Load-Balanced Migration of Social Media to Content Clouds[*]

Xu Cheng
School of Computing Science
Simon Fraser University
British Columbia, Canada
xuc@cs.sfu.ca

Jiangchuan Liu
School of Computing Science
Simon Fraser University
British Columbia, Canada
jcliu@cs.sfu.ca

ABSTRACT

Social networked applications have been more and more popular, and have brought great challenges to the network engineering, particularly the huge demands of bandwidth and storage for social media. The recently emerged content clouds shed light on this dilemma. Towards the migration to clouds, partitioning the social contents has drawn significant interests from the literature. Yet the existing works focus on preserving the social relationship only, while an important factor, user access pattern, is largely overlooked.

In this paper, by examining a large collection of YouTube video data, we first demonstrate that partitioning the network entirely based on social relationship would lead to unbalanced partitions in terms of access. We further analyze the role of social relationship in the social media applications, and conclude that user access pattern should be taken into account and social relationship should be dynamically preserved. We formulate the problem as a constrained k-medoids clustering problem, and propose a novel Weighted Partitioning Around Medoids (wPAM) solution. We present a dissimilarity/similarity metric to facilitate the preservation of the social relationship. We compare our solution with other state-of-the-art algorithms, and the preliminary results show that it significantly decreases the access deviation in each cloud server, and flexibly preserves the social relationship.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Algorithms, Measurement, Performance

Keywords

Social Network, Content Cloud, Clustering

[*]This research is supported by a Canada NSERC Strategic Project Grant, an NSERC Discovery Grant, an NSERC DAS Grant, and an MITACS Project Grant.

1. INTRODUCTION

Social networked applications and services have been dominating the Web 2.0 world in the recent years. The most popular applications include YouTube[1] for video sharing, Facebook[2] for online social networking, and Twitter[3] for micro-blogging. Besides these representatives, many other user-generated content (UGC) applications have emerged and been developing extremely fast.

Generally, it is difficult if not impossible to predict the impact and the development of any UGC application in advance. The provision of resource is thus a great challenge, because any application is possible to grow to the similar scale to YouTube and Facebook, and any one is possible to fail. The applications have brought great challenges to the network engineering, and social media faces a much greater challenge because of the huge resource demands of bandwidth and storage. The recently emerged content cloud service sheds light on this dilemma. Clouds provide "pay-as-you-go" service that allows designers to start small and easy to grow big [1]. A migration that moves the current non-cloud contents, particularly video contents, into clouds is essential for the benefit of the social networked application's development and competition.

To move such application into content clouds, one of the most important steps is to partition the social media contents and assign them into a number of cloud servers. Different from traditional web contents that are isolated, social media contents have connections among each other, and thus the partition is non-trivial. There are some existing works trying to solve this problem, e.g., SNAP [2] and SPAR [15]. Aiming at preserving the social relationship, they are quite effective tools to partition the social network.

However, considering the cloud scenario, one important factor, user access pattern, is largely overlooked in the previous studies. If the videos are assigned into cloud servers entirely based on social relationship, a possible result is that some servers hold many very popular videos but some hold many very unpopular ones, even if the load-balance in terms of the video number is considered. By the evidence of our measurement study on a large collection of YouTube video data, we demonstrate the existence of this phenomenon, which would cause great problem in cloud computing, from the perspective of network engineering. Specifically in the client/server architecture scenario, some cloud servers with many popular videos would be accessed much more fre-

[1]http://www.youtube.com
[2]http://www.facebook.com
[3]http://twitter.com

quently than the other servers with many unpopular videos, and this behavior would decrease the utilization of cloud computing [1]. Even worse, one cloud server may not handle the computation and transmission of the intensive data, while workloads of others are extremely low. The unbalanced partition would also lead to network traffic problems, thus further degrading the quality of service.

On the other hand, partitioning the social media is not simply isolating the popular videos and evenly distributing the others. Taking YouTube as an example, we analyze the role of social relationship in the application, and argue that the social relationship should be dynamically preserved.

In this paper, we formulate the problem as a constrained k-medoids clustering problem, and present a dissimilarity/similarity metric to facilitate the preservation of the social relationship. We propose a novel Weighted Partitioning Around Medoids (wPAM) algorithm to partition the social networked video repository, focusing on load-balance in terms of access. We evaluate our solution on YouTube data, comparing with state-of-the-art algorithms. The preliminary results show that wPAM achieves extremely low deviation of load in terms of the popularity, and flexibly preserves the social relationship under different requirements.

2. RELATE WORK

Web 2.0 UGC applications are emerging in the recent years, and there have been numerous related measurement studies, particularly on understanding YouTube for video sharing [16][5][16], Facebook, MySpace and LinkedIn for online social networking [12][3], Twitter for micro-blogging [9], and etc. Most of them focus on the social structure, user behaviors, and network usage.

There are some works trying to detect the communities and partition social networks. Newman et al. studied a set of algorithms for discovering community structure [13]. The algorithms iteratively remove edges, identified by "betweenness" measure, from the network to split it into communities. Mishra et al. introduced a new criterion that overcomes the limitations that clusters typically do not overlap, by combining internal density with external sparsity in a natural way [11]. SNAP [2] is a tool for analyzing and partitioning small-world network, and it introduces a series of algorithms trying to maximize the modularity of the graph in a parallel manner. SPAR [15] is a work similar to ours, as it considers the cloud scenario. SPAR replicates linked nodes in the same server, and tries to minimize the replications. But as mentioned, while the social relationship is an important factor to the efficiency of the cloud computing, user access pattern is overlooked in their work. Also, different from their work, we focus on the migration to clouds, and thus the maintenance such as adding and removing nodes and edges, adding and removing cloud servers, is not our focus in this paper.

3. MOTIVATION

In this section, we argue that partitioning the social media entirely based on social relationship is not enough, as user access pattern should be taken into account. By examining YouTube video data, we show the correlation between the social relationship and the user access pattern. We also discuss the role of social relationship in social media applications, and conclude that a dynamic preservation of social relationship is preferred.

3.1 Understanding User Access Pattern

User access pattern is yet to be considered while partitioning the social media. We show strong evidence that partitioning entirely based on social relationship would lead to unbalanced partitions in terms of popularity.

We have crawled YouTube videos and obtained a dataset containing over 40, 000 videos. The methodology of the data collection can be referred to our previous work [5]. We focus on the information of video ID, popularity (the number of views), and IDs of the related videos in the dataset. YouTube video graph is a directed graph, and the dataset only records the top-twenty outgoing related video IDs for each video, yet the incoming videos can be easily found within the dataset, and the number might be much greater than 20. We are particularly interested in the number of views and incoming links.

Figure 1 shows the scatter plot of the video's popularity against the number of incoming links. There is a clear trend that videos with more incoming links have greater number of views. This is because videos with more incoming links have more chances to be accessed through related videos. Yet videos with few incoming links might also be very popular, because our measurement dataset is not the entire YouTube video repository. Nevertheless, videos with many incoming links are mostly popular.

We further study the correlation between the video's popularity and the neighbors' popularity. We calculate the mean of neighbors' views, and plot in Figure 2, which clearly shows the positive correlation. Figure 3 further shows the CDF of the ratio of neighbors' popularity and its own popularity. Most of the videos have the comparable number of views as their neighbors' (ratio between 0.1 and 10). This characteristic indicates that if a video is popular, its neighbors are probably also popular, and vice versa.

In summary, a popular video's social neighbors are probably also popular, and they are likely to be clustered based on social relationship only.

3.2 Social Relationship Is Not Enough

From the measurement above we know that partitioning the social media entirely based on social relationship would lead to unbalanced partitions, i.e., some cloud servers have many very popular videos, but others have many very unpopular videos. This behavior would cause great problems in the content clouds as we discussed.

Figure 4 gives a simple example: 8 nodes, each with a weight in terms of popularity as shown, constitute a social graph, which is divided into two parts. As a result, the number of inter-connections is 2 on the left graph, but the standard deviation of the total weight in each part is as great as 348; while on the right graph that we take popularity into account, although the number of inter-connections is 4, the standard deviation is as small as 23.

3.3 Beyond Social Relationship

It is intuitive to preserve the social relationship when partitioning the social graph, but we raise our question: is social relationship that important? Towards answering this question, we analyze the role that social relationship plays in social media applications.

We take the representative YouTube as an example, and discuss videos' social relationship in particular. Each YouTube video has a list of related videos, which constitute

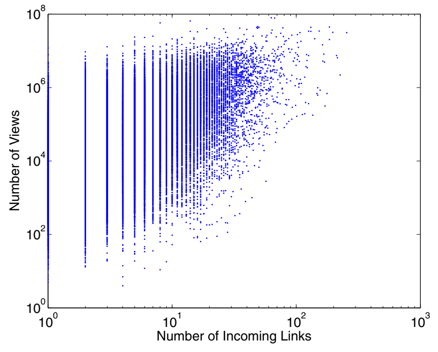

Figure 1: Popularity against incoming links

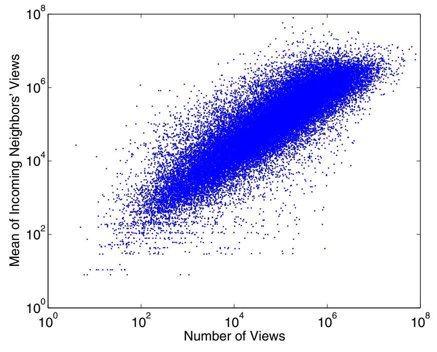

Figure 2: Mean of neighbor view against number of views

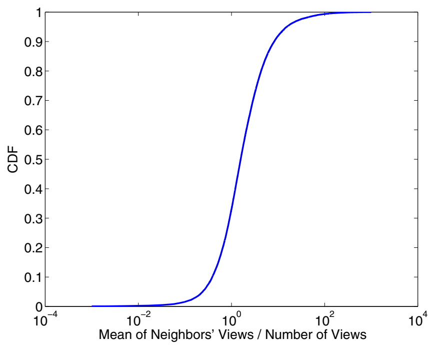

Figure 3: CDF of mean of neighbors' views over number of views

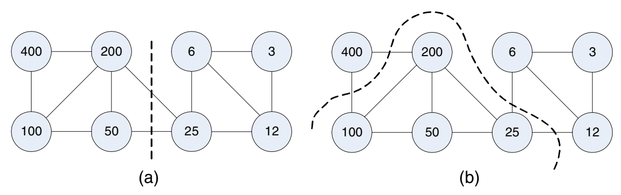

Figure 4: Example of different partitions based on (a) social relationship only, (b) both social relationship and popularity

a social graph. Two videos that have similar titles, tags, descriptions are likely to be linked to each other. The related videos can also be considered as the recommendation from the YouTube system [16]. As studied, one related video is possible to be requested after the user finishes the current video [5][16]. If two related videos are held by the same cloud server, the system can quickly locate the second one to achieve a smooth transition; if the two videos are held by two different cloud servers, this process would take longer, and thus delay would occur. Therefore from this perspective, preserving social relationship is beneficial.

However, the social relationship among videos is not so important that preserving it should be the top priority. As the measurement study shows, about 30% of the overall views are referred from the related recommendation, which is the most important view source [16]. This characteristic indicates that the system's recommendation has a great influence on user's viewing behavior, and thus the system can easily adapt to the situation where related videos are in different cloud servers, e.g., by lowering the rank of the recommendation of the videos that are in different cloud servers. In fact, Zhou et al. suggested utilizing YouTube recommendation to help increase the diversity of video views in aggregation, which encourages users to discover more videos of their interest rather than the popular videos only [16].

Furthermore, a pre-fetching mechanism, where the next video is predicted and being fetched while the user is watching the first video, has been proposed to reduce the startup delay for social media sharing [4]. Considering a system utilizes such a mechanism, sending the two videos at the same time from the two respective cloud servers works better, because it takes advantage of the cloud computing and better utilizes the network resource. Thus from this perspective, breaking the social relationship is beneficial, which seems counter-intuitive but is the case under this circumstance.

In summary, the importance of preserving social relationship is not so significant, and thus it should not be the top priority. Some systems can easily adapt to the situation where related videos are held by different cloud servers, and if the systems utilize a pre-fetching mechanism, it is even better to break the social relationship.

4. PROBLEM STATEMENT

In this section, we first formulate the problem statement, and then introduce a new distance metric used in the problem. We also discuss the details about weight constraint.

4.1 Formulation

Consider a social graph with N nodes $n_1, n_2, \ldots, n_N$, each node n_i has a weight w_i.

We try to partition the nodes into k clusters (cloud servers), $C_1, C_2, \ldots, C_k$. Each cluster C_j has a weight W_j, which is the summation of all the weights of the nodes in cluster C_j, $W_j = \sum_{n_i \in C_j} w_i$.

We suppose there is a representative node o_j in each cluster C_j. We denote $d(n_s, n_t)$ as a distance metric between the two node n_s and n_t. The problem is to find a partition which minimizes

$$E = \sum_{j=1}^{k} \sum_{n_i \in C_j} d(n_i, o_j)$$

subject to

$$|W_{j_1} - W_{j_2}| < \Delta$$

for $j_1, j_2 = 1, 2, \ldots, k$, where Δ is a weight difference constraint.

The problem can be considered as a k-medoids clustering problem with constraint. k-medoids problem is related to k-means problem, except that k-means calculates the mean in each cluster as the center yet k-medoids selects a representative node in each cluster as the center. k-medoids method is more robust than k-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean [6].

4.2 Node Distance – Dissimilarity/Similarity

In the previous studies, several metrics for measuring social networks are used, such as betweenness [13], conductance [11], modularity [2], and number of replicas [15]. Since k-medoids problems generally use Euclidean distance as the metric, we initially considered using integral number of shortest path length between nodes as the metric. After further consideration, we found that the range of the integral path length is however too small. Studies have shown that the shortest path length for YouTube dataset is about 8 [5], and that individuals are separated by six degrees of social contact, known as "six degree of separation" [10]. Furthermore, to compute the path length between two nodes requires the whole knowledge of the social graph, which is very costly.

We thus introduce a new metric, dissimilarity/similarity. It provides a much larger range of calculation than integral number of path length, and computing it only requires the knowledge of the adjacent nodes, which is much more efficient than requiring the information of the whole graph.

We define the similarity metric as follows: consider two nodes n_s and n_t, each has a set of adjacent nodes A_s and A_t. Let $A_s^* = A_s \cup n_s$ and $A_t^* = A_t \cup n_t$. The similarity is calculated as

$$\mathbf{sim}(n_s, n_t) = \frac{|A_s^* \bigcap A_t^*|}{|A_s^* \bigcup A_t^*|}.$$

Different from Jaccard similarity coefficient [7], we include the node itself in the adjacent node set, because we take the relationship between the two target nodes into account as well. Take Figure 5 as an example, both n_s and n_t have four adjacent nodes, yet they are not adjacent on the left while they are on the right. If we do not include the node itself in the adjacent node set, i.e., the similarity is calculated by Jaccard similarity coefficient as $\frac{|A_s \bigcap A_t|}{|A_s \bigcup A_t|}$, the results are 1 and 0.6, respectively, but the right graph is closer to our concept of "similar". By our definition of similarity, the results are 0.67 and 1, respectively.

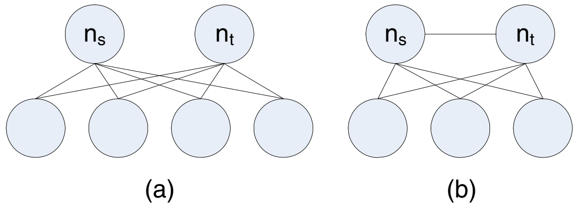

Figure 5: Similarity calculation

To cluster the similar nodes, we require the smaller distance indicating closer nodes, and thus we use dissimilarity as the metric, which is defined as

$$\mathbf{dissim}(n_s, n_t) = 1 - \mathbf{sim}(n_s, n_t).$$

Therefore, the dissimilarity/similarity metric has the unique advantage: we can either use dissimilarity as the metric to preserve the social relationship, or use similarity as the metric to break the social relationship, i.e., to cluster the dissimilar nodes.

4.3 Weight Constraint

In this application, the weight of a node is the number of views of the video, which reflects the possibility of being accessed by the users.

We do not utilize standard deviation for the constraint, because the mathematical formula of the standard deviation would make the problem difficult if not entirely infeasible to solve (we will calculate the standard deviation for the evaluation results). Since it is nearly impossible to decrease the standard deviation to zero, i.e., all the clusters have the same weight, it is much better to use a threshold. In our problem statement, the difference between the total weight of any two clusters should be less than Δ.

Clearly, the smaller the threshold is, the tighter the constraint is, and the less the social relationship will be preserved. Thus there exists a trade-off between social relationship preservation and load-balancing, and we will examine it in the evaluation by testing different value of Δ.

5. SOLUTION

To solve the problem, we propose a wPAM algorithm in this section, and explain the modifications from the original PAM algorithm. We also discuss methods to improve its efficiency and scalability.

5.1 wPAM

Both k-medoids and k-means clustering problems have already been proven to be NP-hard, and a variety of heuristic algorithms have been proposed. PAM (Partitioning Around Medoids) was one of the first k-medoids algorithms introduced [8]. We develop a Weighted PAM algorithm (wPAM) based on PAM, and Algorithm 1 shows the pseudo-code.

Algorithm 1 wPAM

```
Input:
$N$ nodes $n_1, n_2, \ldots, n_N$ with weight $w_1, w_2, \ldots, w_N$,
cluster number $k$,
weight constraint threshold $\delta$.
Output:
a set of $k$ clusters satisfying the weight difference constraint:
$|W_{j_1} - W_{j_2}| < k \cdot \delta$.
Method:
(1) arbitrarily choose $k$ nodes as the initial representative nodes (medoids);
(2) repeat
(3)     assign each remaining node with weight $w$, to the first nearest cluster with weight $W_j$, if satisfying $W_j + w < \overline{W} + \delta$, where $\overline{W}$ is the average weight of all the clusters;
(4)     randomly select a non-representative node, $n'$;
(5)     compute the total cost, $S = E' - E$, of swapping representative node, $o_j$, with $n'$;
(6)     if $S < 0$ then swap $o_j$ with $n'$ to form the new set of $k$ representative nodes;
(7) until no change;
```

Step (3) is the only modification from the original PAM, and there are two major differences. First, in PAM, the distance is between the target node and the representative node in the cluster. We found that there is a great chance that the target node has no mutual neighbors with any representative nodes, and thus using dissimilarity/similarity metric would not get the nearest one. To address the problem, we not only calculate the distance between the target node and the representative node, but also the distance between the target node and the set of nodes that are in the cluster. Specifically, the target node is n_t, and the set of nodes in the cluster is A_s; we apply the distance calculation in Section 4.2, and further calculate the mean of the two distances.

Second, in PAM, each node is just assigned to the nearest cluster. Since we have weight constraint, when assigning the node, we need to ensure that the resulted cluster's weight will not exceed the maximum. In the perfect case, each cluster has a weight $\overline{W}$; considering all clusters but one have weight $\overline{W} + \delta$, which is the maximum result from the algorithm, the last cluster thus has weight $\overline{W} - (k-1) \cdot \delta$. The difference between the last cluster and the others is $k \cdot \delta = \Delta$, which is the weight difference threshold in the problem statement.

Step (5) is the same as the original algorithm, but we explain it in detail here. This step calculates the difference of E if a current representative node is replaced by a non-representative node,

$$E' - E = \sum_{n_i \in C_j} d(n_i, n') - \sum_{n_i \in C_j} d(n_i, o_j).$$

If this total cost is negative, then n_j is replaced or swapped with n' since the actual E would be reduced; if the total cost is positive, the current representative node, n_j, is considered acceptable, and nothing is changed in the iteration. In our experiment in Section 6, the number of iteration turns out to be under 5 in most cases.

5.2 Improving Efficiency

The complexity of iteration in PAM is $O(k(n-k)^2)$, where k is the number of clusters and n is the number of nodes. Therefore, it does not scale well for large dataset.

A sampling-based method, CLARA (Clustering LARge Applications), was introduced to deal with larger dataset [8]. The idea is taking a small portion of the data into consideration, choosing medoids from the sample using PAM. If the sample is selected fairly random, it should closely represent the original dataset, and the medoids chosen would likely be similar to those that would have been chosen from the whole dataset. The complexity of each iteration becomes $O(ks^2 + k(n-k))$, where s is the size of the sample.

CLARANS (Clustering Large Applications based upon RANdomized Search) was further proposed [14]. Unlike CLARA, CLARANS draws a sample with some randomness in each step of the search, and it has been experimentally shown to be more effective than both PAM and CLARA.

Both CLARA and CLARANS are based on PAM, and thus we can utilize either method based on our wPAM algorithm to solve the problem without any further modification. More details about CLARA and CLARANS can be referred to [8] and [14], respectively.

6. EVALUATION

We implement wPAM algorithm to evaluate its performance. For weight difference constraint, we test Δ being $0.1 \cdot \sum W$, $0.01 \cdot \sum W$ and $0.001 \cdot \sum W$, and the number of clusters (cloud servers) k being 4, 8, 16, 32, 64 and 128.

To compare, we also implement pLA algorithm in SPAN [2] and SPAR algorithm [15], as well as a random algorithm that assigns each video to a random cluster without considering social relationship. Briefly, pLA algorithm is a greedy aggregation algorithm to merge the nodes/clusters that increases the overall modularity score, and SPAR algorithm assigns each node to a cluster that needs to generate the fewest replicas. More details about the two algorithms can be referred to the papers [2][15]. Because all the algorithms are randomized, we run each algorithm five times and calculate the average results and their standard deviations.

Our top priority is load-balance, and hence we first look at the result of the standard deviation of each cluster's weight. The cluster's weight is the summation of all the videos' weight (popularity) in the cluster, and reflects the possibility of being accessed. The results against the number of clusters are shown in Figure 6. We normalize the result by dividing the total weight of all the videos in the dataset. Note that both axes are in logarithmic scale. pLA algorithm performs worst, because it tends to detect the popular community, and moreover, pLA does not consider load-balance in terms of the video number in each cluster, and thus the weight deviation is extremely high when the number of clusters is small. Random algorithm performs well, because videos are randomly assigned and thus load-balance is partly achieved. SPAR algorithm performs worse than random algorithm. The reason is the same as that of pLA, yet SPAR strictly balances the number of videos in each cluster, and thus it performs better than pLA. When the weight difference constraint is loose and the number of clusters is small, wPAM algorithm performs similar to SPAR, and becomes better when the number of clusters increases. When the constraint is tight, wPAM outperforms all the compared algorithms, as the green and blue solid lines shown in the figure, thanks to the strict constraint when clustering (step 3) in Algorithm 1.

Supposing the system requires to preserve the social relationship as much as possible, we run wPAM using dissimilarity metric. To test how well the social relationship is preserved in the resulted partition, we generate 10 test cases, each contains 10,000 YouTube video viewing transactions, which are used in work [4]. If two consecutive videos are held by two different cloud servers, we define it as one *transition*, and thus the less the number of transitions is, the better the social relationship is preserved. Because SPAR replicates all the linked nodes (called slave node) of a master node, the next video of a master video is always in the same server. Therefore, only if the next video of a slave video is in a different server, the number of transitions increases. We calculate the percentage of the transitions by dividing the total number 10,000.

Figure 7 shows the results. Not surprisingly, random algorithm performs worst, because it does not take social relationship into account at all. On the other hand, SPAR performs best, since replicas are always in the same cloud server. Our wPAM performs similar to pLA, and with looser weight difference constraint, it performs better as expected, but the difference is not big. Although SPAR outperforms wPAM in terms of preserving social relationship, it requires much more space to store the replicas, about 4 to 12 times more than wPAM for different values of k. Nevertheless, wPAM achieves a rather good performance, even though our priority is load balance.

One advantage of wPAM is that we can break the social relationship to cluster the dissimilar videos, using similarity as the metric, which other algorithms are not capable to do. Figure 8 shows the result using the same test cases as above, and we compare it with the same random algorithm. As a result, when the number of clusters increases, wPAM outperforms random algorithm, and nearly achieves 100% transitions, i.e., the next video is always in a different server.

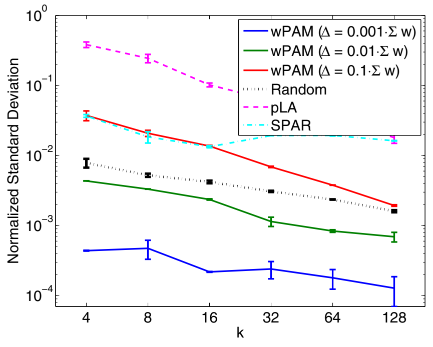

Figure 6: Comparison of normalized weight deviation

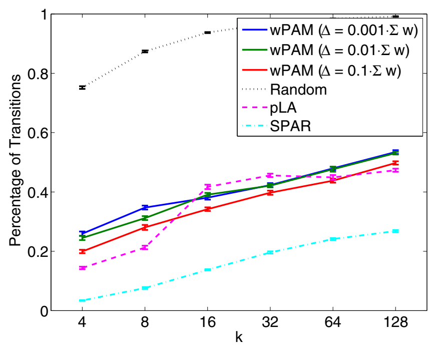

Figure 7: Comparison of percentage of transitions (preserving social relationship)

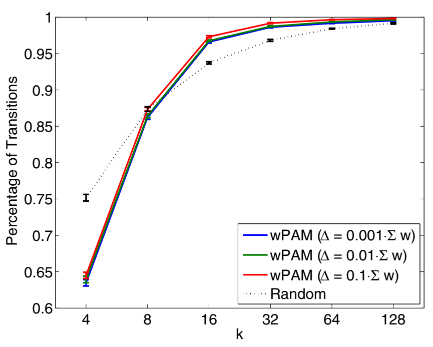

Figure 8: Comparison of percentage of transitions (breaking social relationship)

Looser weight difference constraints do not significantly improve the result. Therefore, considering the performance in terms of load-balance, we suggest using tight constraint when the system requires to break the social relationship.

7. CONCLUSION AND FUTURE WORK

In this paper, we took user access pattern into account in the problem of partitioning social contents, especially social media, in the cloud scenario. By examining YouTube video data, we have demonstrated that partitioning social media content entirely based on social relationship leads to unbalanced access, which would cause great problems to cloud computing. We concluded that a dynamic preservation of social relationship is preferred, by analyzing the role of social relationship in the social media applications. We have formulated the problem as a constrained k-medoids clustering problem, and proposed wPAM algorithm, which significantly decreases the deviation of access in each cluster, and flexibly preserves the social relationship.

Although preliminary results show positive performance of wPAM, we are doing more experiments to validate, in particular with larger datasets. We are also trying to find a more efficient solution to solve the problem. In addition, we are doing the same research on other types of social networked applications, such as Twitter and Facebook. In this paper, we only considered the client/server (C/S) architecture, yet peer-to-peer (P2P) architecture worths investigating as well. The difference between the two scenarios is that, from the server perspective, unpopular videos would be more likely to be requested in P2P, as opposite to C/S, because popular videos are more likely to be shared among peers. P2P scenario is more complicated, as the key issue is how to assign weight to each video to apply the wPAM algorithm.

8. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical report, University of California at Berkeley, 2009.

[2] D. A. Bader and K. Madduri. SNAP, Small-world Network Analysis and Partitioning: An open-source parallel graph framework for the exploration of large-scale networks. In *Proc. of IPDPS*, 2008.

[3] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *Proc. of IMC*, 2009.

[4] X. Cheng and J. Liu. NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing. In *Proc. of INFOCOM*, 2009.

[5] X. Cheng, J. Liu, and C. Dale. Understanding the Characteristics of Internet Short Video Sharing: A YouTube-based Measurement Study. *IEEE Transactions on Multimedia*, 2011.

[6] J. Han and M. Kamber. *Data Mining Concepts and Techniques (2nd Edition)*. Morgan Kaufmann, 2006.

[7] Jaccard Index. http://en.wikipedia.org/wiki/Jaccard_index.

[8] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, 1990.

[9] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *Proc. of WWW*, 2010.

[10] S. Milgram. The Small World Problem. *Psychology Today*, 2(1):60–67, 1967.

[11] N. Mishra, R. Schreiber, I. Stanton, and R. Tarjan. Clustering Social Networks. In *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 56–67. Springer Berlin / Heidelber, 2007.

[12] A. Nazir, S. Raza, and C.-N. Chuah. Unveiling Facebook: A Measurement Study of Social Network Based Applications. In *Proc. of IMC*, 2008.

[13] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69(2):026113, 2004.

[14] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. of VLDB*, 1994.

[15] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The Little Engine(s) That Could: Scaling Online Social Networks. In *Proc. of SIGCOMM*, 2010.

[16] R. Zhou, S. Khemmarat, and L. Gao. The Impact of YouTube Recommendation System on Video Views. In *Proc. of IMC*, 2010.