|| Volume 2 || Issue 3 || OCTOBER 2017 || ISSN (Online) 2456-0774 ||
INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH
AND ENGINEERING TRENDS

Cost Management for Online Networks on Geo-Distributed Cloud

Sayyad Toufique Shafik[1], Sayyad Nadeem Shafiq[2]
PG Student (ME-CSE), Everest Educational Society's College of Engineering and Technology, Aurangabad[1]
BE (CSE), DIEMS, Aurangabad, India[2]
toufique9011@gmail.com[1],snadim232@gmail.com[2]

Abstract – Geo-distributed clouds provide an intriguing platform to deploy online social network (OSN) services. To leverage the potential of clouds, a major concern of OSN providers is optimizing the monetary cost spent in using cloud resources while considering other important requirements, including providing satisfactory quality of service (QoS) and data availability to OSN users. In this paper, we study the problem of cost optimization for the dynamic OSN on multiple geo-distributed clouds over consecutive time periods while meeting predefined QoS and data availability requirements. We model the cost, the QoS, as well as the data availability of the OSN, formulate the problem, and design an algorithm named. We carry out extensive experiments with a large-scale real-world Twitter trace over 10 geo-distributed clouds all across the US. Our results show that, while always ensuring the QoS and the data availability as required, can reduce much more one-time cost than the state-of-the-art methods, and it can also significantly reduce the accumulative cost when continuously evaluated over 48 months, with OSN dynamics comparable to real-world cases.

Keywords *– Cloud computing, online social network, optimization models and methods, performance analysis and evaluation.*

I INTRODUCTION

Internet services today are experiencing two remarkable changes. One is the unprecedented popularity of online social networks (OSNs), where users build social relationships and create and share contents with one another. The other is the rise of clouds. Often spanning multiple geographic locations, clouds provide an important platform for deploying distributed online services. Interestingly, these two changes tend to be combined. While OSN services often have a very large user base and need to scale to meet demands of users worldwide, geo distributed clouds that provide Infrastructure-as-a-Service can match this need seamlessly and provide tremendous resource and cost efficiency advantages. Infinite on-demand cloud resources can accommodate the surges of user requests; flexible pay-as-you-go charging schemes can save the investments of service providers; and cloud infrastructures also free service providers from building and operating one's own data centres. Indeed, a number of OSN services are increasingly deployed on clouds, e.g., Sonico, CozyCot, and Lifeplat [2]. Migrating OSN services toward geographically distributed clouds must reconcile the needs from several different aspects.

First, OSN providers want to optimize the monetary cost spent in using cloud resources. For instance, they may wish to minimize the storage cost when replicating users' data at more than one cloud, or minimize the intercloud communication cost when users at one cloud have to request the data of others that are hosted at a different cloud. Moreover, OSN providers hope to provide OSN users with satisfactory quality of service (QoS). To this end, they may want a user's data and those of her friends to be accessible from the cloud closest to the user, for example. Last but not least, OSN providers may also be concerned with data availability, e.g., ensuring the number of users' data replicas to be no less than a specified threshold across clouds.

Addressing all such needs of cost, QoS, and data availability is further complicated by the fact that an OSN continuously experiences dynamics, e.g., new users join, old users leave, and the social relations also vary. Existing work on OSN service provisioning either pursues least cost in a single site without the QoS concern as in the geo-distribution case or aims for least inter-data-centre traffic in the case of multiple data centres without considering other dimensions of the service [20], e.g., data availability. More importantly, the models in all such work do not capture the monetary cost of resource usage and thus cannot fit the cloud scenario. There are some works on cloud-based social video, focusing on leveraging online social relationships to improve video distribution, which is only one of the many facets of OSN services; most optimization research on multicloud and multi-data-centre services is not for OSN [8][18]. They fail to capture the OSN features such as social relationships and user interactions, and thus their models are not applicable to OSN services. In this paper, we study the problem of optimizing the monetary cost of the dynamic, multicloud-based OSN while ensuring its QoS and data availability.

II MODELS

Targeting the OSN service over multiple clouds, we begin with identifying the types of costs related to cloud resource utilization: the storage cost for storing users' data, the

intercloud traffic cost for synchronizing data replicas across clouds, the redistribution cost incurred by the cost optimization mechanism itself, and some underlying maintenance cost for accommodating OSN dynamics. We discuss and approximate the total cost of the multicloud OSN over time. Afterwards, we propose a vector model to capture the QoS of the OSN service, show the features of this model, and demonstrate its usage. Finally, we model the OSN data availability by linking it with the number of each user's data replicas.

A. System Settings

Clouds and OSN users are all geographically distributed. Without loss of generality, we consider the single-master–multi-slave paradigm [10] Each user has only one master replica and several slave replicas of her data, where each replica is hosted at a different cloud. When signing in to the OSN service, a user always connects to her master cloud, i.e., the cloud that hosts her master replica, and every read or writes operation conducted by a user goes to her master cloud first. We assume the placement of OSN users' replicas follows the social locality scheme . Observing that most activities of an OSN user happen between the user and her neighbors (e.g., friends on Facebook or followees on Twitter), this scheme requires that a user's master cloud host a replica (either the master or a slave) of every neighbor of the user. This way, every user can read the data of her friends and her own from a single cloud, and the intercloud traffic only involves the write traffic for maintaining the consistency among a user's replicas at different clouds.

Social locality has multifold advantages: Given that there are often many more reads than writes in an OSN service [12], it can thus save a large proportion of the intercloud traffic; this scheme also incurs a much lower storage consumption than full replication in that the full replication requires every cloud to maintain a data replica for every user. Note that for a user with one master and slaves, a write on this user's data always incurs corresponding intercloud writes to maintain consistency. We consider eventual consistency in our work and assume issues such as write conflicts are tackled by existing techniques.

B. Modeling the Storage and the Intercloud Traffic Cost

OSN is commonly abstracted as a social graph, where each vertex represents a user and each edge represents a social relation between two users [21]. We extend this model by associating three distinct quantities with every user.

1) A user has a *storage cost*, which is the monetary cost for storing one replica of her data (e.g., profile, statuses) in the cloud for one billing period.

2) Similarly, a user has a *traffic cost*, which is the monetary cost during a billing period because of the intercloud traffic. As mentioned earlier, due to social locality, in our settings the intercloud traffic only involves writes (e.g., posting tweets, leaving comments). We do not consider *intra*cloud traffic, no matter read or write, as it is free of charge [1], [3].

3) A user has a sorted list of clouds for the purpose of QoS.

C. Modeling the Redistribution Cost

An important part of our cost model is the cost incurred by the optimization mechanism itself, which we call the *redistribution* cost. We generally envisage that an optimization mechanism is devised to optimize the cost by moving data across clouds to optimum locations, thus incurring such cost. The redistribution cost is essentially the intercloud traffic cost, but in this paper we use the term intercloud traffic to specifically refer to the intercloud write traffic for maintaining replica consistency, and treat the redistribution cost separately. We expect that the optimization is executed at a per-billing period granularity (e.g., per-month) for the following reasons.

First, this frequency is consistent with the usual charging unit for a continuously running and long-term online service. The OSN provider should be enabled to decide whether to optimize the cost for each billing period, according to her monetary budget and expected profit, etc. Also, applying any cost optimization mechanism too frequently may fail the optimization itself. At the time of writing this paper, the real-world price of intercloud traffic for transferring some data *once* is quite similar to that of storing the same amount of data for an entire billing period [1][3]. As a result, moving data too frequently can incur more redistribution cost that can hardly be compensated by the saved storage and intercloud traffic cost. Without loss of generality, we assume that the optimization mechanism is applied only *once* at the *beginning* of each billing period, i.e., the redistribution cost only occurs at the beginning of every billing period.

III ALGORITHM

Our cost optimization problem is an integer programming (IP) problem. The huge user population of real-world OSN services translates into a huge number of decision variables, and the NP-hardness of our problem makes it impossible to be efficiently solved by existing general-purpose IP solvers. We thus seek practical heuristics. We propose, an optimization algorithm that iteratively swaps the roles of master and slave replicas on different clouds to reach the optimal placement.

A. Observations

Our algorithm is inspired by the following three observations when swapping a master replica and a slave replica of a user. In what we call a role-swap process, the master replica becomes a slave replica and the slave becomes the master.

B. Our Algorithm: Cosplay

Inspired by the above three observations, we employ a series of role-swaps to maximize the total cost reduction while maintaining data availability and ensuring QoS requirements. Our algorithm follows a greedy approach in using role-swaps and requiring that *every* applied role-swap

reduce cost. The more cost reduction each role-swap has and the more role-swaps are applied, the more total cost reduction we can achieve. Note that our algorithm computes a better placement, and it does not physically manipulate data. When our algorithm terminates, data are role-swapped or moved (in the case of redistribution) from existing locations to new locations in order to implement the new placement output by our algorithm.

Algorithm 1: ***isSingleFeasible***(c_{ui}, c_{uj})

Data: c_{ui}, c_{uj}: u's ith and jth most preferred cloud
$\vec{Q}_l$, $\vec{Q}_u$: the QoS lower and upper bounds
$\vec{q}$: the current QoS of the placement
begin
  if $i < j$ **then** // c_{ui} is more preferred than c_{uj}
    for *each* $k \in [i, j-1]$ **do**
      if $\vec{q}[k] - \frac{1}{|V|} < \vec{Q}_l[k]$ **then**
        return $false$;
  else
    for *each* $k \in [j, i-1]$ **do**
      if $\vec{q}[k] + \frac{1}{|V|} > \vec{Q}_u[k]$ **then**
        return $false$;
  return $true$;

Algorithm 2: ***isDoubleFeasible***$(c_{ui}, c_{uj}, c_{vi}, c_{vj})$

Data: c_{ui}, c_{uj}: u's ith and jth most preferred cloud
c_{vi}, c_{vj}: v's ith and jth most preferred cloud
begin
  if ***isSingleFeasible***(c_{ui}, c_{uj}) **then**
    adjustQoS(c_{ui}, c_{uj});
    if ***isSingleFeasible***(c_{vi}, c_{vj}) **then**
      adjustQoS(c_{uj}, c_{ui});
      return $ture$;
    else
      adjustQoS(c_{uj}, c_{ui});
  if ***isSingleFeasible***(c_{vi}, c_{vj}) **then**
    adjustQoS(c_{vi}, c_{vj});
    if ***isSingleFeasible***(c_{ui}, c_{uj}) **then**
      adjustQoS(c_{vj}, c_{vi});
      return $ture$;
    else
      adjustQoS(c_{vj}, c_{vi});
  return $false$;

IV EVALUATIONS

We carry out extensive evaluations by placing real-world Twitter data over 10 clouds all across the US. We demonstrate significant one-time and accumulated cost reductions with compared to existing approaches, while always ensuring QoS and data availability requirements. By varying the experimental settings, we also investigate the complex tradeoff among cost, QoS, and data availability.

A. Data Preparation

By crawling Twitter, we acquired a social graph of 321 505 users with 3 437 409 social relations, all within the US. For each user, we also have her geographic location and tweets. We select 10 cities as cloud locations: Seattle (WA), Palo Alto (CA), Orem (UT), Chicago (IL), San Antonio (TX), Lansing (MI), Alexandria (LA), Atlanta (GA), Ashburn (VA), and New York (NY). We sort the clouds for each user based on geographic distance. We extract 48 monthly OSN snapshots from our Twitter data, from March 2006 to February 2010. Fig. 1 shows the monthly growth rates of the 48 graphs. Based on real-world cloud prices, we calculate the storage cost and the traffic cost of each user in each month. We use Exponentially Weighted Moving Average to do the cost estimation at the beginning of each month.

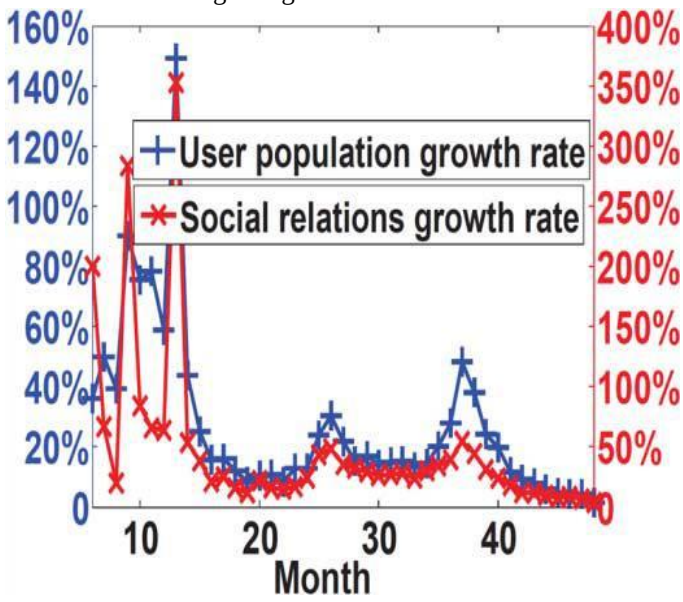

Figure 1 Monthly growth rate

B. Experimental Settings

We run two groups of evaluations. In the first group, with our largest February 2010 social graph as input, we compare the costs and the QoS' of the data placements produced by the *greedy* method, the *random* method, SPAR, METIS, and cosplay. We also investigate how the costs are influenced by the data availability requirement and by the QoS requirement. We ensure social locality for all approaches for fair comparison.

The *greedy* method places every user's master on her first most preferred cloud. The *random* method assigns a user's master to a cloud randomly. For SPAR, we implement it ourselves, and we treat each social relation between two users as an edge creation event and create a random permutation of all events to produce the edge creation trace as input, following the method suggested in. For METIS, there is an open-source implementation from its authors. We use its option of minimizing the interpretation communication. We use each user's storage cost plus her traffic cost as the vertex size (in METIS' terminology) to create its input. For, cosplay we use the *greedy* method to produce an existing placement. We vary the number of most preferred clouds that users use to place masters, and we also vary the QoS and the data availability requirements. We have 10 clouds sorted for every user. Besides the 10-clouds case, we also compare the cases when each user uses her 2, 4, 6, and 8 most preferred clouds for master placement. We vary the data availability

requirement by iterating R from 0 to 9. We set $\beta = 1$, reflecting the fact that the cost of moving some data across clouds once is similar to that of storing the same data in the cloud for one month.

In the second group of evaluations, with the inputs of our 48 monthly OSN snapshots with real-world costs and the other 48 monthly snapshots with estimated costs, we focus on the *continuous* cost reduction that can be achieved by cosplay, compared to the *greedy* method. For each month, we run *greedy* on the former, representing the real-world common practice of placing user's data on the closest cloud for lowest access latency. We run cosplay on the former to show the "ideal" cost reduction, assuming we know the exact costs of each user for each month at the beginning of every month. We also run cosplay on the latter, where replica locations are adjusted according to the estimated costs of each user, to show the effectiveness of our estimation approach. Note that cosplay runs only *once* at the beginning of every month. When new users join the system during a month, each user is still placed by the *greedy* method. When only using *greedy*, the total cost for each month is the sum of the storage and the intercloud traffic cost, plus the maintenance cost

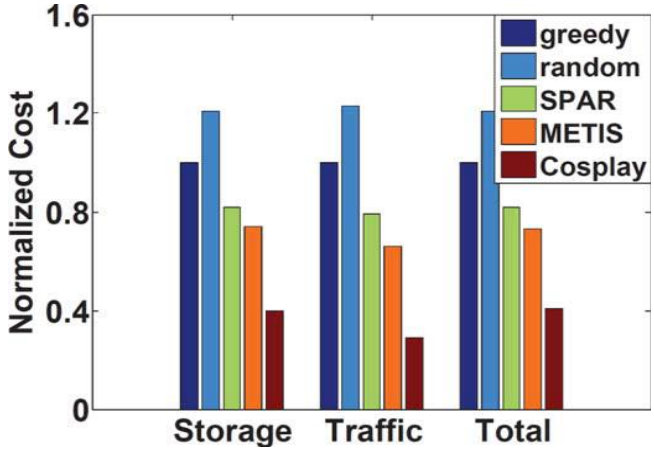

Figure 2 Cost comparison (I)

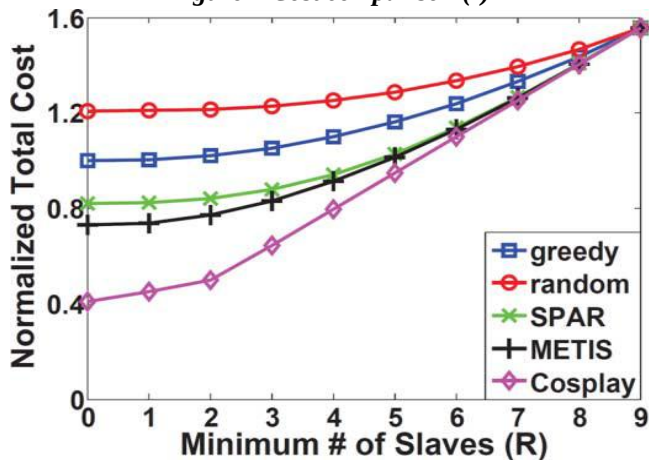

Figure 3 Cost comparison (II)

C. Evaluation Results

In the figures, the cost of every placement is normalized as the quotient of the placement divided by the standard cost, where the standard cost is the cost of the *greedy* placement with R=0 The storage cost is normalized by the standard storage cost, the intercloud traffic cost and the redistribution cost are normalized by the standard intercloud traffic cost, and the total cost is normalized by the standard total cost.

1) One-Time Cost Reduction: The *greedy* placement has moderate cost compared to random. Users who are geographically close to one another tend to have similar sorted lists of clouds. Thus, *greedy* can assign local users to the same nearby cloud, and random tends to straddle local social relations across clouds. SPAR has less cost than greedy and random but more than METIS, indicating that minimizing the number of replicas cannot necessarily minimize the actual cost. Outperforms all others with total cost reductions of 59%, 66%, 50%, and 44%, compared to greedy, random, SPAR, and METIS, respectively.

V RELATED WORK

We contrast our work in this paper with existing work in the following three categories.

Optimizing OSN Services:

For OSN at a single site, using distributed hash to partition the data across servers [5][19] potentially leads to poor performance. Recent work proposes maintaining social locality to address this issue: SPAR minimizes the total number of slave replicas while maintaining social locality for every user; S-CLONE maximizes the number of users whose social locality can be maintained, given a fixed number of replicas per user. For OSN across multiple sites, some propose selective replication of data across data centers to reduce the total inter-data-center traffic [20], and others propose a framework that captures and optimizes multiple dimensions of the OSN system objectives simultaneously [15].

The work in and does not have the concern of QoS as in our geo-distribution case. Besides, the cost models in all the aforementioned existing work, except [15], do not capture the monetary expense and cannot fit the cloud scenario, while [15] and [20] do not explore social locality to optimize the multi-data-center OSN service.

Graph (Re) Partitioning:

The graph partitioning problem divides a weighted graph into a given number of partitions in order to minimize either the weights of edges that straddle partitions or the interpretation communication volume while balancing the weights of vertices in each partition [7]. The repartitioning problem additionally considers the existing partitioning, minimizing the migration costs while balancing vertex weights. State-of-the-art solutions for such problems include METIS [17] and Scotch. Although similar in the sense of partitioning, the problem studied in this paper has fundamental difference from the classic graph (re)partitioning problems. First, classic problems have no notion of social locality, QoS,

and data availability, which makes these algorithms inapplicable to geo-distributed OSNs. Second, classic problems generally define a balance constraint, which is not necessary in the multicloud scenario because each cloud is supposed to provide "infinite" resources on demand.

Optimizing Multicloud Services:

The work most related to OSN services may be those on social media that leverage online social relationships to improve media delivery. Volley [8] finds out the best data center for each data item based on access interdependencies, the identity, and timestamp of data access, while balancing storage capacity across data centers; PNUTS [16] proposes selective replication at a per record granularity to minimize replication overhead and forwarding bandwidth while respecting policy constraints. A substantial body of literature studies cloud resource pricing and allocation [18], request mapping, and content routing in the multicloud or multi-data-center scenario. Although our work also focuses on multicloud services, OSN is unique in data access patterns (i.e., social locality), making this group of existing work inapplicable to our scenario.

VI CONCLUSION

In this paper, we study the problem of optimizing the monetary cost spent on cloud resources when deploying an online social network service over multiple geo-distributed clouds. We model the cost of OSN data placement, quantify the OSN quality of service with our vector approach, and address OSN data availability by ensuring a minimum number of replicas for each user. Based on these models, we present the optimization problem of minimizing the total cost while ensuring the QoS and the data availability. We propose as our algorithm. By extensive evaluations with large-scale Twitter data, is verified to incur substantial cost reductions over existing, state-of-the-art approaches. It is also characterized by significant one-time and accumulated cost reductions over 48 months such that the QoS and the data availability always meets predefined requirements.

REFERENCES

[1] Amazon, Seattle, WA, USA, "Amazon EC2 pricing" [Online]. Available: http://aws.amazon.com/ec2/pricing/

[2] Amazon, Seattle, WA, USA, "Case studies" [Online]. Available: http:/ /aws.amazon.com/solutions/case-studies/

[3] Microsoft, Redmond, WA, USA, "Pricing overview: http://www.windowsazure.com/en-us/ pricing/details/

[4] Facebook, Menlo Park, CA, USA, "Facebook Newsroom" [Online].Available: http://newsroom.fb.com

[5] GitHub, "Twitter/Gizzard" [Online]. Available: http://github.com/twitter/gizzard

[6] Hub Spot, Inc., Cambridge, MA, USA, "State of the Twittersphere,"2010.

[7] A. Abou-Rjeili and G. Karypis, "Multilevel algorithms for partitioning power-law graphs" in Proc. IPDPS, 2006, pp. 1–10.

[8] S. Agarwal et al., "Volley: Automated data placement for geo-distributed cloud services" in Proc. NSDI, 2010, p. 2.

[9] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution" in Proc. SIGKDD, 2006, pp. 44–54.

[10] J. Baker et al., "Megastore: Providing scalable, highly available storage for interactive services" in Proc. CIDR, 2011, pp. 223–234.

[11] A. L. Barabasi, "The origin of bursts and heavy tails in human dynamics," Nature, vol. 435, no. 7039, pp. 207–211, 2005.

[12] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behaviour in online social networks" in Proc. IMC, 2009, pp. 49–62.

[13] H. Chun et al., "Comparison of online social relations in volume vs interaction: a case study of cyworld" in Proc. IMC, 2008, pp. 57–70.

[14] H. Hu and X. Wang, "Evolution of a large online social network.

[15] L. Jiao, J. Li, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services" in Proc. IEEE INFOCOM, 2014, pp. 28–36.

[16] S. Kadambi et al., "Where in the world is my data?" in Proc. VLDB, 2011, pp. 1040–1050.

[17] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs" SIAM J. Sci. Comput., vol. 20, no. 1, pp. 359–392, 1999.

[18] A. Khanafer, M. Kodialam, and K. PN Puttaswamy, "The constrained ski-rental problem and its application to online cloud cost optimization" in Proc. IEEE INFOCOM, 2013, pp. 1492–1500.

[19] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system" Oper. Syst. Rev., vol. 44, no. 2, pp. 35–40, 2010.

[20] G. Liu, H. Shen, and H. Chandler, "Selective data replication for online social networks with distributed datacenters" in Proc. IEEE ICNP, 2013, pp. 1–10.

[21] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks" in Proc. IMC, 2007, pp. 29–42.