Multi-Objective Data Placement for Multi-Cloud Socially Aware Services

Lei Jiao*, Jun Li†, Wei Du*, Xiaoming Fu*
*University of Göttingen, Germany, †University of Oregon, USA
*{jiao,du,fu}@cs.uni-goettingen.de, †lijun@cs.uoregon.edu

***Abstract*—Socially aware services often have a large user base and data of users have to be partitioned and replicated over multiple geographically distributed clouds. Choosing in which cloud to place data, however, is difficult. Effective data placements entail meeting multiple system objectives, including reducing the usage of cloud resources, providing good service quality to users, and even minimizing the carbon footprint, while facing critical challenges such as the interconnection of social data, the conflicting requirements of different objectives, and the customized multi-cloud data access policies.**

In this paper, we study multi-objective optimization for placing users' data over multiple clouds for socially aware services. We build a model framework that can accommodate a range of different objectives, and based on this model we formulate the optimization problem. Leveraging graph cuts, we propose an optimization approach that decomposes our original problem into two simpler subproblems and solves them alternately in multiple rounds. We carry out evaluations using a large group of real-world geographically distributed users with realistic interactions, and place users' data over 10 clouds all across the US. We demonstrate results that are significantly superior to standard and *de facto* methods in all objectives, and also show that our approach is capable of exploring trade-offs among objectives, converges fast and scales to a huge user base.

I. Introduction

Internet services that span multiple geographically distributed clouds intrinsically have multiple system objectives, including budgeting the monetary expenditure spent on cloud resource usage [1], [2], ensuring the service quality perceived by users (*e.g.*, access latency) [3], [4], and even reducing the carbon footprint of the service [5], [6], to name a few. Key to meeting many of such objectives is at which cloud to place the data accessed by each user. For example, different clouds may charge different prices for the same amount of resource consumption, have different proximity to users, and emit different amounts of carbon for the same workload, *i.e.*, they have different carbon intensities. Data placement determines how the workload of a service is distributed over clouds, and thus affects various aspects of the service's performance.

The data placement problem is particularly challenging for multi-cloud services that are *socially aware*, where users build social relationships and share contents with one another, as reflected by Online Social Network (OSN) services and many non-OSN services with social components [7]. Fig. 1 illustrates how one such service is provided at distributed clouds to serve users at different locations, where every user can access every cloud and users form an OSN via online friendships. The challenges for optimizing data placement for such services mainly manifest themselves as follows:

First, because of social relations and interactions of users in a socially aware service, the data of every user are interconnected with data of some others, and data placement on the basis of individual users probably cannot yield optimal results. With heavy interactions between online friends [8], [9], for example, it would be better to have their data placed closely. Ideally, if the data of a user and the data of her friends are always co-located at the same cloud, the user can access her friends' data without going to another cloud, and thus save the additional delay and traffic associated with further operations. This feature distinguishes social data placement from content placement in conventional content distribution networks, where data are independently delivered to users.

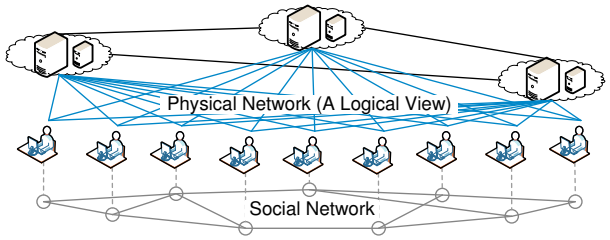


Fig. 1: A Multi-Cloud Socially Aware Service

Moreover, the diverse objectives for data placement are often intertwined and even contradictory, and may not be satisfied simultaneously [5]. The social nature, for example, requires friends' data to be placed closely. For low access latency, data accessed by each user—including those of her friends and her own—are preferred to be located at the clouds close to a user. However, to reduce the monetary expense or the carbon footprint, it is beneficial to place more users' data at clouds that are cheaper in price or more efficient in carbon intensity. A data placement approach thus needs to be capable of seeking trade-offs among multiple objectives.

Further complicating the problem are the multi-cloud master-slave paradigm [2], [10] and the multi-cloud access policies [11]. When the data of every user have a master replica and multiple slave replicas, as is often the case in many services, these replicas contribute differently to system objectives. The location of a user's master contributes to the write latency perceived by all those who write to this user's data. The locations of a user's slaves contribute to the read latency, and different users may read different replicas of a user. If a data replica is not available at a current cloud, different multi-cloud access policies regulate differently "where" and "how" to obtain the required data from another cloud, thus potentially leading to different optimal data placements.

Unfortunately, most previous optimization research on multi-cloud or multi-data-center services [3]–[6], [12], [13] cannot capture users' social relations and their interactions in socially aware services. There are multiple studies on multi-

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

cloud OSN and social media services [1], [2], [14], [15], but they do not address the carbon issue and are not able to weigh costs of multiple dimensions. Furthermore, except the work in [2], [14], little has been done to investigate multi-cloud data placement in the context of the master-slave paradigm—which is very common in reality—and multi-cloud access policies.

In this paper, we capture the multi-objective data placement problem by building a model framework that generalizes to a large variety of system objectives of the multi-cloud socially aware service. We address the aforementioned challenges by proposing a novel approach that leverages the graph cuts technique [16]–[18]. We verify our models and approach via extensive evaluations driven by large-scale real-world data.

Starting with modeling the carbon footprint, the service quality, the inter-cloud traffic of the socially aware service, as well as the reconfiguration cost incurred by changing a given data placement to another, we generalize our models to cover a wide range of system objectives of different dimensions, allowing them to be treated within a common framework. The data placement problem is thus about finding best locations for each user's master and slave replicas in order to minimize the total cost. An interesting observation of our models is that the cost of every dimension of a socially aware service can be naturally cast into one or both of the two parts: the unary cost that depends on the locations of replicas of an individual user, and the pairwise cost that depends on the locations of replicas of a pair of users, *i.e.*, one user who conducts read and write operations and the other user whose data are read or written.

Our core contribution is a novel approach that solves the data placement problem. Intuitively, the unary cost and the pairwise cost correspond to vertices and edges of a graph respectively, motivating us to connect our data placement problem that is centered around cost minimization with the problem of finding the minimum cut of a graph, and to solve the former via solving the latter with the help of the graph cuts technique. Towards this end, we propose to decompose our original problem into two subproblems and solve them alternately in multiple rounds. In one subproblem, given the locations of all slaves, we identify the optimal locations of all masters by iteratively cutting the corresponding graphs. In the other subproblem, we place all slaves given the locations of all masters, where we find that the optimal locations of each user's slaves are independent and a greedy scheme that takes account of all objectives can usually be sufficient. Our approach achieves good data placements overall. On one hand, to the best of our knowledge, the state-of-the-art graph-cut technique guarantees the best solutions; on the other hand, our greedy scheme can empirically achieve results close to the theoretical optimum. By applying it separately to each community of the entire user base, our approach further scales to a huge user population. Doing so may degrade the optimization performance, but only moderately due to the community structure of users in socially aware services.

With 107,734 users interacting over 2,744,006 social relations that span all across the US, we perform data placement over 10 distributed clouds. Our evaluations demonstrate the following results: (1) While there is no other approach known to us that optimizes the multi-objective data placement for multi-cloud socially aware services, our approach significantly outperforms several standard and *de facto* practices, such as random and greedy placement, in all objective dimensions including carbon, distance and traffic in a variety of settings; (2) Our model can be easily tuned to achieve different trade-offs among multiple objectives, and to help decide whether a specific optimization outcome is worth the effort of conducting this optimization; (3) Our approach converges fast, *e.g.*, the first 3 iterations reach approximately 97% of the total cost reduction that can be achieved; (4) Our approach scales, *e.g.*, by partitioning the user base into 4 communities and applying our approach independently to each community, we obtain a speedup of 4.5 in execution time with the optimization performance degraded only slightly by 6%.

II. Model Formulation

We now introduce the system settings of the multi-cloud socially aware service, model its different objectives, and formulate the optimization problem of users' data placement based on this model.

A. Settings and Notations

We target a multi-cloud socially aware service as in Fig. 1, where each cloud is located in a different geographic region and each user has replicas of her data stored in the clouds. We consider the single-master-multi-slave paradigm [10], [19], where every user has one replica as a master and multiple replicas as slaves. Each replica is stored in one cloud. The cloud that hosts a user's master replica is the user's master cloud, and those that host a user's slave replicas are the user's slave clouds. Central to user interactions are the read and write operations between users. We focus on the *number* of reads and writes in our model.

A service with data partitioned and replicated across clouds often follows some multi-cloud access policies about "where" and "how" to obtain the required data from a remote cloud if they are unavailable at a local cloud. This happens when, *e.g.*, a user accesses the service via the web and her read request is directed by DNS to a cloud, but the data requested turns out to be at a different cloud. We handle such access policies in this paper as follows. "Where" is captured in our model by the function $z_{u,v}$, which selects a cloud out of user v's master and slave clouds by any given policy in order to serve user u's requests that access v's data. Note that $z_{u,v}$ only applies to read operations, since write operations are always executed in master clouds, with propagations to slaves afterwards. "How" is captured as either a relay mode or a redirect mode [11]. The former means the local cloud reads the data from another cloud and then returns the data to the user. The latter means the local cloud redirects the user to another cloud and lets the user retrieve the data on her own. Due to space limitation and in order to demonstrate our model's capability of addressing objectives of multiple dimensions, we only focus on the relay case (while the redirect case has fewer objective dimensions, as will be discussed in Section II-B3). One can easily apply our methodologies presented in this paper to the redirect case.

Before going into details, we introduce the notations that are used throughout the rest of this paper. We use u, v to denote users and i, j to denote clouds. The decision variables to be solved are $\mathbf{m}_u$ and $\mathbf{s}_{u,l}$, $l = 1, ..., \mathsf{k}$, $\forall u$, indicating the location of u's master (*i.e.*, the ID of u's master cloud) and those of u's k slaves (*i.e.*, the IDs of u's k slave clouds), respectively. k is the number of slave replicas each user has. r_u and w_u denote the number of reads and that of writes conducted by user u on her own data. r_{uv} and w_{uv} denote the number of reads and that

of writes conducted by user u on user v's data. $\mathbb{N}_u$ is the set of user u's neighbors, where two users are considered neighbors if and only if there exists at least one read or write operation between them. $z_{u,v}(\mathbf{m}_u, \mathbf{m}_v, \mathbf{s}_{v,1}, ..., \mathbf{s}_{v,\mathsf{k}})$ is the function as stated above, returning the ID of the selected cloud according to a given policy. $\delta(x, y)$ is a binary function that returns 1 if $x \neq y$, and 0 otherwise.

B. Modeling Socially Aware Service

1) Modeling Carbon Footprint: The total carbon footprint of the service depends on the workload of each cloud, and also depends on the carbon intensity of the region where a cloud is located. A user reads and writes her own data at her master cloud, as we assume a user is hosted by her master cloud since she signs in to the service, and all of a user's writes are eventually propagated from her master cloud to all her slave clouds for consistency. When a user u reads another user v's data, the cloud determined by $z_{u,v}$ serves such reads. When u writes v's data, the writes go to v's master cloud for execution, and further propagate to all of v's slave clouds.[1]

Let ε be the energy consumption of a single read or write operation and e_i be the carbon intensity of the region where cloud i is located. We write the total carbon footprint as

$$\sum\nolimits_u D_u^c + \sum\nolimits_u \sum\nolimits_{v \in \mathbb{N}_u} V_{u \to v}^c, \tag{I}$$

where $D_u^c = D_u^{c\prime} + D_u^{c\prime\prime}$, with $D_u^{c\prime} = \varepsilon \mathsf{e}_{\mathbf{m}_u}(\mathsf{r}_u + \mathsf{w}_u + \sum_{v \in \mathbb{N}_u} \mathsf{w}_{vu})$, representing the carbon at u's master cloud incurred by all of u's reads and writes on her own data and all the writes conducted by u's neighbors on u's data, and $D_u^{c\prime\prime} = \sum_{l=1}^{\mathsf{k}} (\varepsilon \mathsf{e}_{\mathbf{s}_{u,l}}(\mathsf{w}_u + \sum_{v \in \mathbb{N}_u} \mathsf{w}_{vu}))$, representing the carbon at all of u's slave clouds incurred when all writes to u are propagated to her slaves, and $V_{u \to v}^c = \varepsilon \mathsf{e}_{z_{u,v}} \mathsf{r}_{uv}$ refers to the carbon at the cloud $z_{u,v}$ when u reads her neighbor v.

2) Modeling Operation Distance: We define the *operation distance* of a service as the total geographic distance traveled by all reads and writes occurred in this service, and we use this notion as a measure of service quality. For example, if a user issues a read request and this request is forwarded due to data absence at her master cloud, then the distance traveled by this read is the distance between the user and her master cloud plus the distance between her master cloud and the cloud that this request is forwarded to. [2] All operations issued by a user firstly go to her master cloud, and then if the data required by some operations are not available there, such operations continue to travel to the destination clouds which have the required data.

The operation distance is thus calculated as follows, where $\mathsf{d}_{\cdot,\cdot}$ is the distance between a user and a cloud, or between two clouds:

$$\sum\nolimits_u D_u^d + \sum\nolimits_u \sum\nolimits_{v \in \mathbb{N}_u} V_{u \to v}^d, \tag{II}$$

where $D_u^d = \mathsf{d}_{u,\mathbf{m}_u}(\mathsf{r}_u + \mathsf{w}_u + \sum_{v \in \mathbb{N}_u} (\mathsf{r}_{uv} + \mathsf{w}_{uv}))$ is the user-cloud distance, *i.e.*, the total distance traveled by all of u's operations to go from u herself to her master cloud, and $V_{u \to v}^d = \mathsf{d}_{\mathbf{m}_u, z_{u,v}} \delta(\mathbf{m}_u, z_{u,v}) \mathsf{r}_{uv} + \mathsf{d}_{\mathbf{m}_u, \mathbf{m}_v} \delta(\mathbf{m}_u, \mathbf{m}_v) \mathsf{w}_{uv}$ is the inter-cloud distance, *i.e.*, the sum of the total distance traveled from u's master cloud to the cloud $z_{u,v}$ when u reads v and the total distance traveled from u's master cloud to v's master cloud when u writes v.

3) Modeling Inter-Cloud Traffic: The inter-cloud traffic is incurred by inter-cloud operations. One type of inter-cloud operations are reads and writes that cannot be completed at a local cloud due to absence of data, and are thus executed at a remote cloud. The other type is the writes propagated for replica consistency. However, the amounts of propagated writes are fixed when the inputs (*i.e.*, the number of write operations received by each user and the number of slave replicas of each user) are given, and thus they are out of our optimization framework. Consequently, the inter-cloud traffic under our consideration only exists in the relay mode.

The total amount of inter-cloud traffic reads as follows, assuming t bytes of traffic incurred by a single operation:

$$\sum\nolimits_u \sum\nolimits_{v \in \mathbb{N}_u} V_{u \to v}^t, \tag{III}$$

where $V_{u \to v}^t = \mathsf{t}(\delta(\mathbf{m}_u, z_{u,v}) \mathsf{r}_{uv} + \delta(\mathbf{m}_u, \mathbf{m}_v) \mathsf{w}_{uv})$ is the sum of the inter-cloud traffic incurred when u reads and writes v.

4) Modeling Reconfiguration Cost: Different from carbon, operation distance, and inter-cloud traffic, all of which are associated with one data placement, we introduce the *reconfiguration cost* as a measure of the cost incurred by changing one data placement to another. While there may be many ways to define the reconfiguration cost, here we focus on the number of affected users, *i.e.*, those whose masters change locations.

Let $\mathbf{m}'_u$ denote the location of u's master in the initial data placement, and $\mathbf{m}_u$ denote its location in the optimal one. The reconfiguration cost is

$$\sum\nolimits_u D_u^r, \tag{IV}$$

where $D_u^r = \delta(\mathbf{m}'_u, \mathbf{m}_u)$ simply calculates whether u's master remains at the cloud where it used to be. We will extend this definition of the reconfiguration cost later in Section III-C.

C. Generalizing Models

We generalize (I) to the *intra-cloud cost* that is incurred at all clouds. The intra-cloud cost is calculated by replacing $\varepsilon \mathsf{e}_i$ in (I) with α_i, where we think of α_i as a property (that can be of any dimension) only specific to cloud i. The intra-cloud cost can represent the following metrics:

- The carbon footprint as in (I), if $\alpha_i = \varepsilon \mathsf{e}_i$;
- The electricity fees, if $\alpha_i = \varepsilon \mathsf{q}_i$, where q_i is the per unit electricity price of the region where cloud i is located;
- The fees of using Virtual Machines (VMs) in the clouds, if $\alpha_i = \mathsf{T}\mathsf{p}_i/\mathsf{M}$, where T is the time period during which all operations under consideration are executed, p_i is the price per VM per time unit at cloud i, and M is the number of operations that one VM accommodates during T.

We also generalize $\sum_u D_u^d$ to the *user-cloud cost* that is incurred between all user-cloud pairs. Replacing $\mathsf{d}_{u,i}$ with $\alpha_{u,i}$, a property only specific to user u and cloud i, enables the user-cloud cost to represent the following metrics:

- The operation distance between users and clouds, *i.e.*, $\sum_u D_u^d$ as in (II), if $\alpha_{u,i} = \mathsf{d}_{u,i}$;
- The reconfiguration cost as in (IV), if $\alpha_{u,i} = \delta(\mathbf{m}'_u, i)/\mathsf{G}$, where $\mathsf{G} = \mathsf{r}_u + \mathsf{w}_u + \sum_{v \in \mathbb{N}_u} (\mathsf{r}_{uv} + \mathsf{w}_{uv})$;

[1] Issues out of the scope of this paper include write conflicts resolution [10]. We assume such issues are addressed by existing techniques, which does not affect our work as long as all writes are eventually executed.

[2] Aligning with eventual consistency, we assume a write operation returns to the user as soon as it is completed on the maser replica [19]. Therefore, the operation distance of a write does not involve propagations.

- The total network delay or hop counts between users and clouds, if $\alpha_{u,i} = \mathsf{x}_{u,i}$, where $\mathsf{x}_{u,i}$ is the network delay or hop count between user u and cloud i.

We finally generalize $\sum_u \sum_{v \in \mathbb{N}_u} V^d_{u \to v}$ to the *inter-cloud cost* that is incurred between all pairs of clouds. It is calculated by using $\alpha_{i,j}$ that is specific to a pair of clouds i, j to replace $\mathsf{d}_{i,j}$ in $\sum_u \sum_{v \in \mathbb{N}_u} V^d_{u \to v}$. The inter-cloud cost can represent the following metrics:

- The operation distance between clouds, *i.e.*, $\sum_u \sum_{v \in \mathbb{N}_u} V^d_{u \to v}$ as in (II), if $\alpha_{i,j} = \mathsf{d}_{i,j}$;
- The inter-cloud traffic as in (III), if $\alpha_{i,j} = \mathsf{t}$;
- The total inter-cloud network delay or hop counts, similarly.

Our generalized models potentially cover a wide range of metrics that can be used as system objectives of the multi-cloud socially aware service, going beyond the concerns of cloud customers, *i.e.*, socially aware service providers in our case. The electricity fee is, for instance, not of the interest of cloud customers as they do not directly pay for it, but rather a concern of cloud or data center operators who can also leverage our models to seek optimization. One can explore our models to express even more metrics in the multi-cloud environment.

D. Optimization Problem

Putting all objectives together, we minimize the following cost function:

$$\sum_u D_u(\mathbf{m}_u, \mathbf{s}_{u,1}, ..., \mathbf{s}_{u,\mathsf{k}}) + \sum_u \sum_{v \in \mathbb{N}_u} V_{u \to v}(\mathbf{m}_u, \mathbf{m}_v, \mathbf{s}_{v,1}, ..., \mathbf{s}_{v,\mathsf{k}}), \quad \text{(V)}$$

where $D_u = D^c_u + D^d_u + D^r_u$, $V_{u \to v} = V^c_{u \to v} + V^d_{u \to v} + V^t_{u \to v}$. The most important feature of this formulation is that it consists of two terms: D_u that only depends on the locations of a single user's data (her master and slaves), and $V_{u \to v}$ that depends on the locations of a pair of neighboring users' data (the master of user u who conducts operations and the master and slaves of user v who receives operations). We refer to the former as the unary cost, as it contains the decision variables of only one user, and the latter as the pairwise cost, as it contains the decision variables of a pair of users.

Note that (V) is equivalent to the sum of the three types of costs defined in Section II-C, with $\alpha_i = \varepsilon \mathsf{e}_i$, $\alpha_{u,i} = \mathsf{d}_{u,i} + \delta(\mathbf{m}'_u, i)/\mathsf{G}$, and $\alpha_{i,j} = \mathsf{d}_{i,j} + \mathsf{t}$. Due to the generality of the three types of costs, solving our optimization problem implies solving a class of problems with a variety of system objectives.

One can always associate a weight with each objective, or with each of the three types of costs. For the ease of presentation, we do not write these weights in all our formulas in this paper. Service providers can tune these weights by standard approaches in order to seek trade-offs among objectives based on their own requirements.

As we establish a multi-objective optimization problem, the major constraint that our problem is subject to is no co-location of a user's replicas at a common cloud, which reads as $\mathbf{m}_u \neq \mathbf{s}_{u,l}, \mathbf{s}_{u,l} \neq \mathbf{s}_{u,l'}$, where $l, l' = 1, ..., \mathsf{k}, l \neq l', \forall u$. Note that we do not regard cloud capacity as a constraint for the following reasons: (1) From a cloud customer's perspective, a cloud can provide "infinite" resources on demand; (2) By assuming an unlimited capacity of each cloud, we aim at a lower bound of the total cost possible with our framework.

III. Data Placement Optimization

This section describes how we solve the data placement problem. The direct optimization of (V) is intractable due to its NP-hardness. Our general idea is thus to develop heuristics that seek good approximate solutions.

A. Overview of Our Approach

We have the following insights into the problem of (V).

Insight 1: The difficulty in optimizing (V) roots partially in that the decisions of placing master replicas and slave replicas affect each other.

Insight 2: Our problem can be potentially connected to the minimal s-t cut problem [20]. Naively, consider two clouds and a social network, and let's assign users to clouds. We can add edges to the social graph by connecting every user with every cloud, regarding one cloud as the s terminal and the other as the t terminal. The weight of the edge between any two users is the pairwise cost of assigning them to different clouds, and the weight of the edge between a user and a cloud is the unary cost of assigning this user to this cloud. Consequently, by finding the minimal s-t cut of this graph, we find the set of edges with the minimum total cost, where each edge between a user and a cloud in this cut set represents the optimal assignment of the user to the corresponding cloud.

Motivated by these two insights, we decompose our data placement problem into the following two subproblems.

Master Replicas Placement: The first subproblem is placing users' master replicas given the placement of users' slave replicas, which is formulated as minimizing

$$\sum_u D_u(\mathbf{m}_u) + \sum_u \sum_{v \in \mathbb{N}_u} V_{u \to v}(\mathbf{m}_u, \mathbf{m}_v). \quad \text{(VI)}$$

Leveraging a more sophisticated version of iterative minimal s-t cuts [16]–[18] as will be described in Section III-B, we can identify the optimal assignments of users' masters to clouds.

Slave Replicas Placement: The second subproblem is placing users' slave replicas given the placement of users' master replicas, which is formulated as minimizing

$$\sum_u D_u(\mathbf{s}_{u,1}, ..., \mathbf{s}_{u,\mathsf{k}}) + \sum_u \sum_{v \in \mathbb{N}_u} V_{u \to v}(\mathbf{s}_{v,1}, ..., \mathbf{s}_{v,\mathsf{k}}). \quad \text{(VII)}$$

As will be shown in Section III-C, minimizing the total cost incurred by all users' slaves can be achieved by independently minimizing the cost incurred by each user's slaves.

Overall, our approach consists of solving the two subproblems of minimizing (VI) and (VII) alternately via fixed-point iterations, as in Fig. 2. Starting with an initial placement of all masters and slaves, we solve the two subproblems iteratively to reduce the total cost and to improve the solution of each subproblem, until no further cost reduction is possible or until an expected number of iterations are executed.

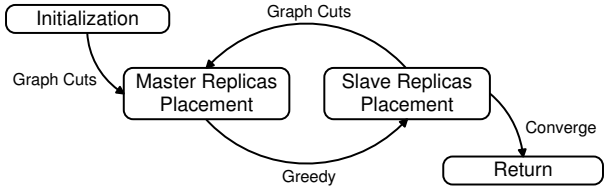


Fig. 2: Our Fixed-Point Iteration Approach

B. Solving Master Replicas Placement

The basic idea is iterating cloud pairs and finding the minimal s-t cut for each pair [18]. Every s-t cut represents the optimal assignments of involved masters to a pair of clouds. The algorithm keeps iterating cloud pairs to update the assignments until the total cost of the assignments of all masters cannot be reduced any more.

Fig. 3 visualizes how this works. Users, *i.e.*, their master replicas, are in the middle while at the top and the bottom are the clouds. An edge between two users means the two users have interactions. An edge between a user and a cloud indicates that the user's master is placed at that cloud. Initially, every user is connected to a cloud arbitrarily. Selecting a cloud pair, *e.g.*, the blue ones in this figure, it constructs a graph by connecting both clouds to every user who is connected to one of the two clouds, as in the left part of Fig. 3, and assigns an appropriate weight to each edge in this constructed graph. The weight of the edge between a user u and a cloud i is computed based on $D_u(i)$, reflecting how much u wants her master to be placed at i, while the weight of the edge between two users u and v is computed based on $V_{u\to v}(i,j)+V_{v\to u}(j,i)$, reflecting how much u and v want their masters to be placed at i and j, respectively. We can always find the minimal s-t cut of the constructed graph by the max-flow min-cut algorithm [20]. The right part of Fig. 3 marks the cut edges by double dashes, where other edges of the constructed graph that are not in the cut set are not shown. The algorithm continues by selecting another cloud pair, *e.g.*, a blue one and the red one, constructs a graph by adding edges, and calculates the minimal s-t cut for the optimal assignments of involved users to these two clouds, and so on.

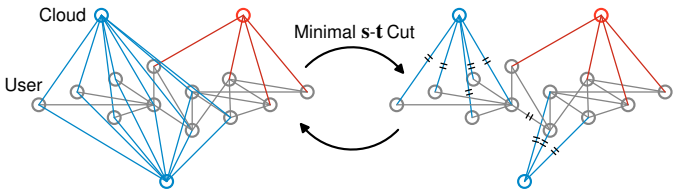


Fig. 3: Using Graph Cuts to Place Master Replicas

C. Solving Slave Replicas Placement

Noticing that we have $\sum_u \sum_{v\in\mathbb{N}_u} V_{u\to v}(\mathbf{s}_{v,1},...,\mathbf{s}_{v,\mathsf{k}}) = \sum_u \sum_{v\in\mathbb{N}_u} V_{v\to u}(\mathbf{s}_{u,1},...,\mathbf{s}_{u,\mathsf{k}})$, we apply it and transform (VII) to

$$\sum_u \left(D_u(\mathbf{s}_{u,1},...,\mathbf{s}_{u,\mathsf{k}}) + \sum_{v\in\mathbb{N}_u} V_{v\to u}(\mathbf{s}_{u,1},...,\mathbf{s}_{u,\mathsf{k}})\right)$$

which implies that slave data placements for different users are independent, given the locations of all users' masters. Thus, we can solve this subproblem by finding the optimal placement of the slave replicas of each user separately.

To place a slave of a user, a greedy method finds the current best cloud for this replica, places the slave there, and continues to place the next slave of the same user until all k slaves are placed. The cost incurred by placing place u's lth ($2 \le l \le \mathsf{k}$) slave at cloud i ($i \neq \mathbf{m}_u$, $i \neq \mathbf{s}_{u,l'}$, $l' = 1,...,l-1$) is $\gamma_{u,i} = \alpha_i(\mathsf{w}_u + \sum_{v\in\mathbb{N}_u} \mathsf{w}_{vu}) + \beta_{u,i}|_{z'''_{v,u}} - \beta_{u,i}|_{z''_{v,u}}$, where $\beta_{u,i} = \sum_{v\in\mathbb{N}_u}((\alpha_{z_{v,u}} + \alpha_{\mathbf{m}_v,z_{v,u}}\delta(\mathbf{m}_v,z_{v,u}))\mathsf{r}_{vu})$, $\alpha_\cdot$ and $\alpha_{\cdot,\cdot}$ are as in Section II-C, $\beta_{u,i}|_{z'''_{v,u}}$ means calculating $\beta_{u,i}$ by replacing $z_{v,u}$ with $z'''_{v,u} = z_{v,u}(\mathbf{s}_{u,1},...,\mathbf{s}_{u,l-1},i)$, and $\beta_{u,i}|_{z''_{v,u}}$ means calculating $\beta_{u,i}$ by replacing $z_{v,u}$ with $z''_{v,u} = z_{v,u}(\mathbf{s}_{u,1},...,\mathbf{s}_{u,l-1})$. $z'''_{v,u}$ returns the selected cloud if u's lth slave slave is placed at cloud i, and $z''_{v,u}$ returns the selected cloud when this slave does not exist in the system. Therefore, the best cloud for u's lth slave is $i = \arg\min_i \gamma_{u,i}$. When $l = 1$, we naturally do not have $\beta_{u,i}|_{z''_{v,u}}$ and only use $\beta_{u,i}|_{z'''_{v,u}=i}$ in $\gamma_{u,i}$.

An exhaustive method is also possible. Except a user's master cloud, one can select k clouds out of all other clouds to place this user's k slaves, and after checking all the possibilities, place slaves on those where the total incurred cost is minimum. This exhaustive approach always finds the theoretical optimum. We experimentally find that graph cuts with the greedy approach achieves only about 1-2% worse than with the exhaustive one, which indicates that a greedy approach can be empirically sufficient in our case.

Note that $\gamma_{u,i}$ can be extended to include $\delta'_u(i)$ as part of it, where $\delta'_u(i) = 0$ if $i \in \{\mathbf{s}'_{u,1},...,\mathbf{s}'_{u,\mathsf{k}}\}$ and $\delta'_u(i) = 1$ otherwise, and $\mathbf{s}'_{u,l}$, $l = 1,...,\mathsf{k}$, are u's slave locations in the initial placement. $\delta'_u(i)$, jointly with (IV), enables us to calculate the total number of moved masters and slaves as the reconfiguration cost.

D. Discussions

Optimality: We believe that the solutions found by our approach are reasonably good, as the two subproblems are either solved by the current state-of-the-art technique of graph cuts or fairly easy to be solved due to the independence among users. To the best of our knowledge, each subproblem is solved to the best that can be achieved to date. Although our approach cannot guarantee the Pareto efficiency theoretically, it is experimentally justified by our evaluations which will be shown in the next section.

Scalability: The most time-consuming part of our approach is calculating the minimal s-t cut by the max-flow min-cut algorithm, which could be computationally expensive for an extremely large user population. However, in socially aware services, users often form communities within which they interact heavily and across which sparsely. We can thus partition the user base into communities by algorithms like METIS [21], and then apply our approach to each community independently and even in a parallel manner.

Regularity: The graph cuts technique that we use requires the pairwise cost to obey the *regularity* property [16]. Translated into our case, it means when solving the master replicas placement problem, $V_{u\to v}(\mathbf{m}_u,\mathbf{m}_v) + V_{v\to u}(\mathbf{m}_v,\mathbf{m}_u)$ must satisfy $V_{u\to v}(i,i) + V_{v\to u}(i,i) + V_{u\to v}(j,j) + V_{v\to u}(j,j) \le V_{u\to v}(i,j) + V_{v\to u}(j,i) + V_{u\to v}(j,i) + V_{v\to u}(i,j)$, $\forall u,v,i,j$. Regularity actually encourages the co-location of the masters of users who have interactions between them—which matches the requirement of social data placement.

We analyze how our case satisfies regularity. To align $V_{u\to v} + V_{v\to u} = V^c_{u\to v} + V^d_{u\to v} + V^t_{u\to v} + V^c_{v\to u} + V^d_{v\to u} + V^t_{v\to u}$ with regularity, we deduce two sufficient conditions: $z_{u,v}(\mathbf{m}_u,\mathbf{m}_v) = z_{u,v}(\mathbf{m}_u)$, $\forall u,v$; $z_{u,v}(\mathbf{m}_u,\mathbf{m}_v) = \mathbf{m}_v$, $\forall u,v$. The regularity is satisfied as long as one of these two conditions holds. The former requires $z_{u,v}$, the selection of one of v's clouds, not depend on which v's master cloud is. This can be implemented by making the selection only out of v's slave clouds, *e.g.*, selecting the slave cloud of v that is closest to u's master cloud. The latter requires $z_{u,v}$ always select v's

master cloud, which is naturally true in a system with no slave replicas (*i.e.*, k = 0). In reality, however, it is possible that $z_{u,v}$ does not necessarily have one of the above two forms, *e.g.*,

$$z_{u,v} = \begin{cases} \mathbf{m}_u, if\ \mathbf{m}_u = \mathbf{m}_v\ or\ \mathbf{m}_u = \mathbf{s}_{v,l}, \exists l \in \{1, ..., \mathsf{k}\} \\ z'_{u,v}(\mathbf{m}_u, \mathbf{m}_v), otherwise \end{cases}, \quad \text{(VIII)}$$

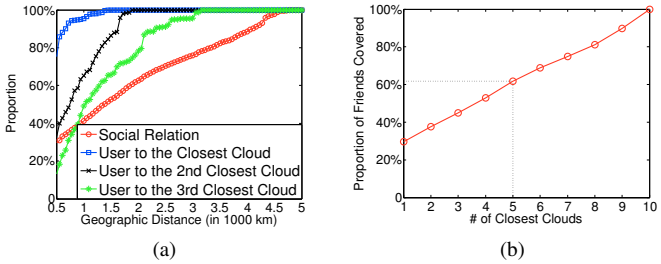
where u always accesses her master cloud if v has a replica co-located there, and accesses another cloud if not. To make $V_{u\to v} + V_{v\to u}$ with $z_{u,v}$ in (VIII) satisfy regularity, we also deduce a sufficient condition: $z'_{u,v}(\mathbf{m}_u, \mathbf{m}_v) = \mathbf{m}_v$, which requires u's requests be always forwarded to v's master cloud if v's data are not found at u's master cloud. However, $z'_{u,v}(\mathbf{m}_u, \mathbf{m}_v) \neq \mathbf{m}_v$ does not tend to cause a problem practically. Note that $V_{u\to v}(i,i) + V_{v\to u}(i,i) + V_{u\to v}(j,j) + V_{v\to u}(j,j) = (\alpha_i + \alpha_j)(r_{uv} + r_{vu})$, and $V_{u\to v}(i,j) + V_{v\to u}(i,j) + V_{u\to v}(j,i) + V_{v\to u}(j,i) = (\alpha_{z'_{u,v}(i,j)} + \alpha_{z'_{u,v}(j,i)})r_{uv} + (\alpha_{z'_{v,u}(i,j)} + \alpha_{z'_{v,u}(j,i)})r_{vu} + (V^d_{u\to v}(i,j) + V^t_{u\to v}(i,j) + V^d_{u\to v}(j,i) + V^t_{u\to v}(j,i) + V^d_{v\to u}(i,j) + V^t_{v\to u}(i,j) + V^d_{v\to u}(j,i) + V^t_{v\to u}(j,i))$. It is obvious that the the latter equation is definitely no smaller than the former if $\alpha_i = \alpha_j$, $\forall i,j$ holds. As discussed in Section II-C, α_i's can capture the carbon intensity, the electricity or VM price. In reality, the difference of each of these quantities at different clouds or regions does not differ by more than one order of magnitude [5], [13]—it is thus easy to tune the weights associated with $V^d_{u\to v}$, $V^t_{u\to v}$ and α_i to make the latter equation large enough, which translates into the practical success of graph cuts no matter what $z_{u,v}$ is used by service providers.

IV. Experimental Evaluation

With real-world data trace as inputs, we run simulations in a variety of realistic settings. We demonstrate that our approach can achieve significantly better results than existing approaches, explore trade-offs among objectives, converge fast, and scale to a huge user base.

A. Data Preparation

Users: We obtained 107,734 users all across the US with 2,744,006 social relations among them by crawling Twitter in a breadth-first manner in 2010. We translate each user's profile location into geographic coordinates (*i.e.*, [latitude, longitude]), enabling us to calculate the geographic distance. Our Twitter graph is a single connected component, and is used as an undirected social graph throughout our evaluations, as in [22]. Fig. 4a is a CDF, showing that about 30% of social relations in our dataset stretch within 500 km and all the rest spread almost uniformly over geographic distance.



(a) (b)

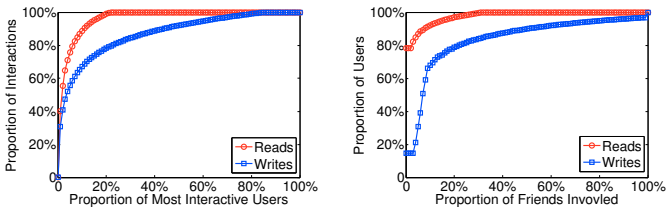
Fig. 4: Geo-Distributions of Users and Clouds

Clouds: According to the geographic distribution of users, we select 10 regions across the US and select a city out of each region as the location of a cloud. Fig. 4a confirms that our clouds are at or near locations with dense user populations, as about 80% of users can find a cloud closest to them with no longer than 500 km. Fig. 4b indicates that 30% of all friends of a user have the same closest cloud as the user. Considering more clouds, we see that, *e.g.*, the 5 closest clouds to a user can include the closest cloud to about 60% of all the user's friends. The default factors for estimating carbon emissions are also reported on a per region basis [23]. Table I lists the cities we select in a carbon ascending order.

TABLE I: Carbon Intensities of Cloud Locations

City	State	eGRID Region [23]	CO_2 (lb/MWh)
Palo Alto	CA	WECC California	658.68
Fall River	MA	NPCC New England	728.41
Seattle	WA	WECC Northwest	819.21
Secaucus	NJ	RFC East	947.42
Ashburn	VA	SERC Virginia/Carolina	1035.87
Miami	FL	FRCC All	1176.61
San Antonio	TX	ERCOT All	1181.73
Atlanta	GA	SERC South	1325.68
Lansing	MI	RFC Michigan	1659.46
St. Louis	MO	SERC Midwest	1749.75

Interactions: It is extremely hard to obtain interaction traces, especially for read operations such as one user browsing another user's profile. Service providers are often reluctant to share such data due to competition and privacy concerns [24]. User interactions differ from other types of cloud workloads in how read and write operations are distributed among users and among users' friends. Recent literature, fortunately, disclosed such features for a small-scale and local OSN [8], [9], making it possible to synthesize *realistic* user interactions. Fig. 5 provides CDF distributions of the operations synthesized among our real-world Twitter users. Without loss of generality, we have precisely scaled the OSN interactions of the smaller, regional OSN to our large-scale, geo-distributed OSN.



(a) (b)

Fig. 5: Distributions of User Interactions

We highlight how we capture the distributions of real-world user interactions and achieve the aforementioned realistic synthesization with our social graph. We first do curve fitting for the distributions of reads and writes among top interactive users, and the distributions of reads and writes among each user's friends reported in [9], and use these fitted curves as inputs. With a given total number of reads and that of writes conducted by all users, we then perform the following steps: (1) Sort all users in the descending order of social degree. With the reads and writes distributions among top interactive users, calculate for each user the number of reads and that of writes that this user conducts to her neighbors. (2) With the sorted users and the distributions of reads and writes among each user's friends, calculate for each user the number of

neighbors that this user reads and writes. (3) For each user, select the specified number (as calculated in the previous step) of neighbors, and for a user's each selected neighbor, assign the number of reads and writes conducted by this user to this neighbor as being proportional to this neighbor's social degree (as in the preferential model [25]). Note that we do not consider users' geographical locations during this procedure, as location does not obviously influence interactions [26].

B. Evaluation Settings

Interaction Workload: As stated in the previous section, we can control the total number of reads and that of writes to produce different workloads while always maintaining the featured distributions of interactions. We use the ratio of the total number of reads over that of writes "R/W" to denote workloads. We evaluate the case of "R/W = 10", to reflect the fact that OSN services have many more reads than writes [9], [14], and the case of "R/W = 1", to investigate how "R/W" may affect the benefits of our approach.

Number of Slaves: We have 10 clouds, and we evaluate 8 cases by iterating the number of slave replicas per user from 1 to 8. The number of slaves per user depends on a lot of factors, *e.g.*, data availability requirements, the monetary budget of the service provider, *etc.* We do not intend to decide what number is the best for a provider. Here what we want to check is whether our solution is better in all objective dimensions than existing approaches for any given number of slaves per user.

Multi-Cloud Access Policies: Leveraging (VIII), we set two policies that we consider would be among the most practical ones. The "Master" policy where $z'_{u,v} = \mathbf{m}_v$, and the "Closest" policy where $z'_{u,v} = \arg\min_i(\mathsf{d}_{\mathbf{m}_u,i})$, $\forall i \in \{\mathbf{m}_v, \mathbf{s}_{v,1}, ..., \mathbf{s}_{v,\mathsf{k}}\}$. The former has been explained in Section III-D. The latter means that, when v does not have a replica at u's master cloud, u's read requests go to v's cloud (either her master cloud or one of her slave clouds) that is closest to u's master cloud for execution. Similarly, we do not seek to decide the best access policy.

Weights of Multi-Objectives: We have four objectives to optimize and thus four weights. We vary the weights to seek trade-offs among objectives, and use the ratio of weights to denote our variations. Note that we have normalized all dimensions of inputs to the same order of magnitude in value, and the ratios reported throughout the evaluations are the ones of weights associated with the normalized inputs.

Figure Settings: Table II summarizes the specific settings corresponding to each figure. "A/F" means that a figure is varying this setting for comparison, and relevant information is available in the figure itself. Note that we do not consider the reconfiguration cost except in Fig. 12, thus the ratios in this table are between three weights rather than four.

TABLE II: Evaluation Settings for Figures

Fig.	R/W	Slave # (k)	Policy	Weights
6, 13	10	A/F	Closest	1:1:1
7, 8, 9	10	A/F	A/F	1:1:1
10	A/F	4	A/F	1:1:1
11	10	4	Closest	A/F
12, 14	10	4	Closest	1:1:1

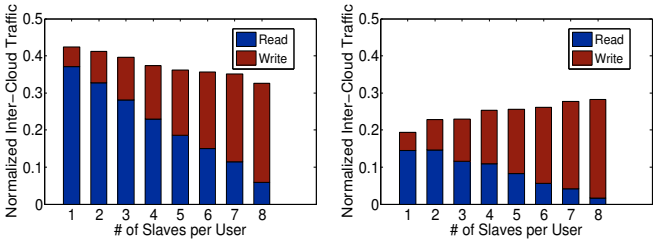
Algorithmic Settings: We use C++ to implement different placement methods. In particular, as for our proposed approach to solve master data placement, we calculate various costs according to our models and feed them to the `gco-v3.0` library [16]–[18], an open source implementation of graph cuts. We invoke this library with the option of α-β-`swap` [18].

C. Evaluation Results

How much benefit can we gain? We compare the data placements produced by our approach with those produced by random placement, the standard practice of distributed databases (*e.g.*, MySQL) and key-value stores (*e.g.*, Cassandra), and by greedy placement, the *de facto* practice of many real-world services [10], [11]. The random approach places each replica of a user randomly at one of the clouds. The greedy approach places a user's master at the closest cloud to that user, and places her k slaves at the other closest k clouds to that user. As in Fig. 2, our approach uses greedy placement at initialization.

Fig. 7 shows the operation distance of different data placements. The distance always drops as a user has more slaves, since data become available at more clouds and more operations can be completed locally or nearby. Greedy beats random because slaves randomly placed at clouds are less likely to benefit friends, due to the locality shown in Fig. 4b. Our approach beats both random and greedy. Across all cases, we save 33%-54% distance when the master policy is applied, and 7%-48% when the closest policy is applied, compared with greedy. We save even more compared with random. The benefit of our approach over others roughly decreases as the slave number increases, because the number of clouds that do not have a user's replica becomes smaller and less room is left for optimization by rearranging replica locations.

Fig. 8 depicts the inter-cloud traffic (including those incurred by the propagated writes) of different data placements. In the random and greedy placements, the amount of traffic does not depend on access policies. With our approach, using different policies as inputs leads to different placements and thus different amounts of traffic. Our data placements have 13%-78% less traffic than others. We dissect the traffic details in Fig. 6 as follows, where we show greedy and our approach with the closest policy as examples. The growth of the number of slaves per user incurs more write traffic to maintain consistency, while the amount of read traffic becomes less due to the increased data availability at more clouds. Overall, random and greedy have the total traffic descend; the traffic of our solutions keeps increasing, as we reduce the read traffic by a large fraction and the write traffic becomes a dominance.



(a) Greedy (b) Ours

Fig. 6: A Closer Look at Inter-Cloud Traffic

Fig. 9 focuses on the carbon footprint. Our approach saves 10%-30% carbon compared to random and greedy. The essential feature that distinguishes carbon from distance and traffic is that both the latter encourage data to be placed closely for optimization, as stated in Section III-D, while carbon does not necessarily favor this, but rather depends on at which

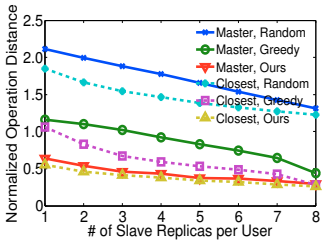


Fig. 7: Operation Distance

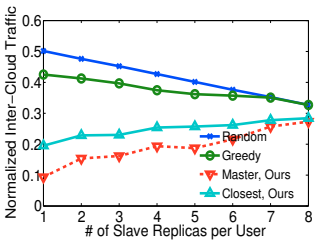


Fig. 8: Inter-Cloud Traffic

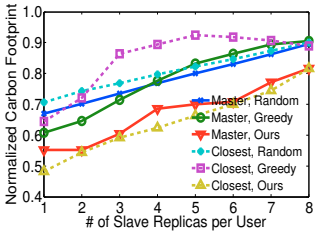


Fig. 9: Carbon Footprint

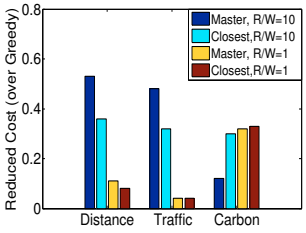


Fig. 10: Influence of Workload

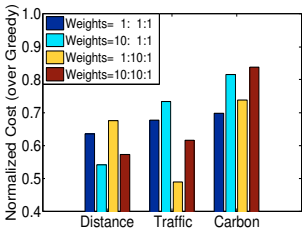


Fig. 11: Tuning Weights

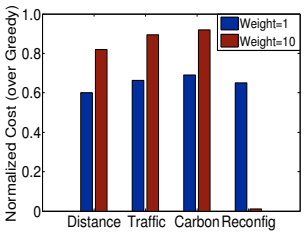


Fig. 12: Reconfiguration

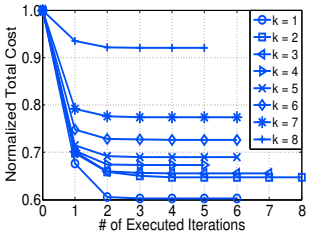


Fig. 13: Convergence

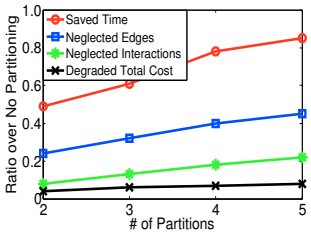


Fig. 14: Scalability

clouds the operations are executed. Random has a steadily growing carbon as the slave number increases, since it tends to span each user's replicas all across the clouds with carbon intensity also spanning a certain range, as in Table I. Greedy's carbon changes up and down since it places data collectively and tends to always use a set of nearby clouds.

How does the workload influence the benefit? Fig. 10 describes how the total number of reads and writes among users may influence the advantages of our approach. For both policies, our approach optimizes distance and traffic more than carbon, when there are more reads than writes, and vice versa. This is normal, because the advantage of our approach lies in optimizing reads, and writes excluding the propagated ones. More writes imply more propagations, leaving the system with less room for optimization, which, in turn, indicates that our approach is more suitable and capable for read-intensive services like socially aware ones and many others.

What are the trade-offs among objectives? Fig. 11 indicates that, by tuning the weight of each objective dimension, one can seek a range of trade-offs without changing any other part of our framework. Here we choose to tune distance and traffic as an example while fixing the weight of carbon. We set the ratio of the distance weight over the traffic weight to be (1) 1:1, (2) 10:1, (3) 1:10, and (4) 10:10. We make the following observations: (2) has a larger distance weight than (1), and thus (2) is smaller in distance, and is in turn larger in traffic and carbon; (3) has a larger traffic weight than (1), and thus (3) is smaller in traffic and larger in distance and carbon; (4) has larger distance and traffic weights than (1), and is thus smaller in these two dimensions and larger in carbon.

Fig. 12 additionally considers the reconfiguration cost, *i.e.*, the total number of moved masters and slaves. By controlling the weight of the reconfiguration cost, one can set it is cheap or expensive to move replicas across clouds. In this figure, we set this weight to be 1 and 10, respectively. One interesting observation is that setting it to be 10 times larger can efficiently prohibit replica movements across clouds. However, this does not prevent optimization, since the 1 case moves a huge many more replicas and only achieves about 30% more optimization than the 10 case.

How fast does our approach converge? Fig. 13 illustrates the total costs of the data placement after each iteration of our approach, varying the number of slaves per user. One iteration includes an execution of graph cuts to solve the master replicas placement and an execution of our greedy method to solve the slave replicas placement. This figure indicates that the most cost reduction is achieved in the first iteration. For all cases, the largest number of required iterations is 8, after which no cost can be reduced any more. Our approach is highly efficient and converges fast. In practice, one can even adopt an early stop strategy, *i.e.*, running 2 or 3 iterations and terminating the algorithm is sometimes already sufficient to achieve a large part of optimization.

How scalable is our approach? Fig. 14 demonstrates the scalability of our approach. We use METIS [21] to partition our original dataset into several partitions, and then apply our approach to each partition independently while neglecting the inter-partition interactions. Doing so saves up to 85% (in the 5-partition case) of the total execution time, and only degrades the total cost of the optimal data placement by less than 8%, compared with running our approach directly on the original dataset. The success roots in the community structure of OSN social relations and interactions, and thus even neglecting 45% social relations and the associated 22% interactions of the original dataset only has a slight influence on the optimization. For real-world data with a stronger community structure, we can expect even less cost degradation.

V. Related Work

We describe existing work in two categories and discuss how our work in this paper differs from them.

Saving energy and carbon of distributed data centers: Qureshi *et al.* [13] first proposed to cut the electricity bill of geo-distributed data centers by leveraging the diversity of electricity prices of different regions. Rao *et al.* [12] defined strategies to distribute user requests to minimize the electricity expense of data centers that consume electricity from multiple markets. Le *et al.* [6] argued that the electricity expense could be reduced by appropriately controlling the proportion of the brown energy and that of the green energy consumed by each

data center. Liu *et al.* [4] minimized the electricity expense by additionally integrating the availability of renewable energy at different regions. Xu *et al.* [3] considered the price diversity of both the electricity and the ISP bandwidth of data centers, and minimized the sum of these two expenditures. Gao *et al.* [5], to the best of our knowledge, did the only work so far of investigating optimizing multiple dimensions of system objectives of distributed data centers or clouds.

All such work targets conventional and non-socially-aware services. Except [5], all assume full data replication across data centers. Even [5] still cannot serve our purpose, as it does not address (1) social relations and user interactions, (2) writes to contents and the maintenance of replica consistency, (3) inter-cloud operations that contribute to QoS and inter-cloud traffic, and (4) the master-slave paradigm that is widely used in reality. Our work, in contrast, captures all such particular features in the context of socially aware services and provides trade-offs among a wide range of metrics via our generalized model framework and a unified, single solution approach.

Placing OSN and social media contents across clouds: Jiao *et al.* [2] studied distributing OSN over multiple clouds in order to minimize the monetary expense of the service, while meeting a pre-specified overall QoS requirement and ensuring social locality for every user. Liu *et al.* [14] chose to replicate data across clouds for selected users instead of replicating for every user, in order to save the total inter-cloud traffic involving reads and writes. Wu *et al.* [1] focused on scaling social media service into geo-distributed clouds with minimum cost over time by exploiting a dynamic optimization framework leveraging social influence among users. Wang *et al.* [15] proposed a hybrid edge-cloud and peer-assisted architecture to serve social videos in order to improve the replication performance and QoS, leveraging the propagation properties of social videos.

This category of work focuses either on the performance of the OSN/social media service [14], [15], or the monetary expense of the service in clouds [1], [2]. Our work differs from such work in that, to our best knowledge, we are the first to include the carbon footprint of socially aware services into consideration, with a complex trade-off among a large variety of related factors such as QoS and inter-cloud traffic. Besides the generality of our models and the unique solution approach, we are also the first to investigate this complicated joint optimization problem in the master-slave paradigm while accommodating customized multi-cloud access policies.

VI. Conclusion

While socially aware services attract billions of users, the need for such services to meet multiple system objectives has become compelling. The unique features of socially aware services that distinguish themselves from other Internet services pose a new problem of optimizing data placement over multiple geographically distributed clouds.

In this paper, we firstly build models that generalize to a variety of system objectives, capturing user interactions, the master-slave paradigm, and the multi-cloud access policies. We then propose an approach with multiple iterations, with each iteration solving master and slave data placement separately, leveraging our finding that the master placement subproblem can be effectively solved via graph cuts. Evaluations with real-world data further show that our approach is not only able to optimize every dimension of the socially aware service, but can also pursue a diversity of trade-offs among objectives, converge fast, and scale to a large user base.

Acknowledgment

This work has been partially sponsored by the EU FP7 IRSES MobileCloud Project (Grant No. 612212).

References

[1] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," in *INFOCOM*, 2012.
[2] L. Jiao, J. Li, T. Xu, and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in *ICNP*, 2012.
[3] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *INFOCOM*, 2013.
[4] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew, "Greening geographical load balancing," in *SIGMETRICS*, 2011.
[5] P. Gao, A. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *SIGCOMM*, 2012.
[6] K. Le, O. Bilgir, R. Bianchini, M. Martonosi, and T. Nguyen, "Managing the cost, energy consumption, and carbon footprint of internet services," in *SIGMETRICS*, 2010.
[7] "Case studies - amazon web services," http://aws.amazon.com/solutions/case-studies/.
[8] C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Zhao, "User interactions in social networks and their implications," in *EuroSys*, 2009.
[9] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Zhao, "Understanding latent interactions in online social networks," in *IMC*, 2010.
[10] Y. Sovran, R. Power, M. Aguilera, and J. Li, "Transactional storage for geo-replicated systems," in *SOSP*, 2011.
[11] N. Tran, M. Aguilera, and M. Balakrishnan, "Online migration for geo-distributed storage systems," in *USENIX ATC*, 2011.
[12] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM*, 2010.
[13] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *SIGCOMM*, 2009.
[14] G. Liu, H. Shen, and H. Chandler, "Selective data replication for online social networks with distributed datacenters," in *ICNP*, 2013.
[15] 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 *ACM Multimedia*, 2012.
[16] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.
[17] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.
[18] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.
[19] J. Baker, C. Bond, J. Corbett, J. Furman, A. Khorlin, J. Larson, J. Léon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing scalable, highly available storage for interactive services," in *CIDR*, 2011.
[20] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.
[21] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1999.
[22] J. 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 *SIGCOMM*, 2010.
[23] "U.s. energy information administration (eia)," http://www.eia.gov/.
[24] M. Mondal, B. Viswanath, A. Clement, P. Druschel, K. Gummadi, A. Mislove, and A. Post, "Defending against large-scale crawls in online social networks," in *CoNEXT*, 2012.
[25] I. Hoque and I. Gupta, "Disk layout techniques for online social network data," *IEEE Internet Computing*, vol. 16, no. 3, pp. 24–36, 2012.
[26] A. Kaltenbrunner, S. Scellato, Y. Volkovich, D. Laniado, D. Currie, E. Jutemar, and C. Mascolo, "Far from the eyes, close on the web: impact of geographic distance on online social interactions," in *SIGCOMM WOSN*, 2012.