DAROS: Distributed User-Server Assignment And Replication For Online Social Networking Applications

Thuan Duong-Ba
School of EECS
Oregon State University
Corvallis, OR 97330, USA
Email: duongba@eecs.oregonstate.edu

Thinh Nguyen
School of EECS
Oregon State University
Corvallis, OR 97330, USA
Email: thinhq@eecs.oregonstate.edu

Duc A. Tran
Department of Computer Science
University of Massachusetts at Boston
Boston, MA 02125, USA
Email: duc@cs.umb.edu

***Abstract*—In this paper we study the problem of assigning users to servers and data replication in a distributed manner for online social networking (OSN) applications. Typical OSN applications such as Facebook and Twitter are built on top of an infrastructure of servers, which handle the users data storage and communications. Thus, for a given user's communication pattern, the loads of the servers depend critically on the assignment of users to servers. A good assignment will reduce the overall load of the system. Furthermore, by replicating data across the servers judiciously, the overall load can also be further reduced. Unfortunately, this optimal assignment and data replication problem is NP-hard. Therefore, we introduce a distributed heuristic algorithm in which the servers perform local computations and exchange information among each other iteratively in such a way that the algorithm converges to a good assignment and replication in terms of reducing the overall system load as well as balancing the loads among the servers. In contrast with a centralized algorithm, a distributed algorithm offers the advantage of balancing the computations among all the servers as well as the ability to naturally adapt to time-varying user's communication patterns. Simulations results show promising performance for the proposed algorithm.**

***Index Terms*—Distributed optimization, social network, large scale distributed systems, data replication**

I. Introduction

For scalability, online social networks (OSNs) such as Facebook, Twitter, Linkedin, etc. employ an underlying infrastructure of servers which handle the communications and data storage on behalf of the users. Specifically, each user is assigned to a primary server. When data is transferred from one user to another, the users do not set up a network connection directly with each other. Instead, their primary servers will perform the data transfer on their behalf. If the two users are assigned to the same primary server, then intercommunication between two servers are not needed. As a result, for a given users communication pattern, the loads of the servers depend critically on the assignment of users to servers. To be concrete, in this paper we consider the following protocol used in a typical scenario of OSN applications *wall posts* in FaceBook.

When a user u goes online, it will (1) issue a read request to its primary server s_u for stored personal data and then request for its friends'information; (2) If its friends are assigned on the same s_u, the read request for friends' messages is simply reading data from the local storage; (3) However, if some of its friends are assigned to different servers, then the primary server forwards the read request to every primary servers of its friends. In response to the request, the friends' primary servers read the data (e.g., the previously posted message by its friends) from their databases, and send them back to s_u. s_u then returns the data to user u.

In this scenario, the server load is incurred by (1) reading data from the database and (2) communication between servers. As an example, let us consider a user u, its two friends, users v and w with their different primary servers s_u, s_v, and s_w, respectively. When user u goes on line, it will read data from its primary server S_u which incurs a read load. Next, S_u send two requests to s_v and s_w to request the updated data from v and w on behalf on u. This incur a communication load on three servers. s_v and s_w then read the updated data from their databases which incur a read load on them. s_v and s_w then return the updated data back to s_u which further incurs a communication cost. We note that the read cost is primarily due to time and computation involved with reading data from disk while the communication cost is primarily due to the bandwidth cost.

The communication cost is reduced if users v and w reside on the same primary server s_u. In that case, no inter-server communication is needed, thus no communication load is incurred. On the other hand, the read load remains the same since three reads from the databases are still required. Thus, to reduce the overall load, it makes sense to assign friends to the same server. However, this is not always possible. One way to reduce the load is to replicate the user data across multiple servers.

Replication. If the data of user v is replicated on the primary server of its friend user u. Then when u goes online,

This work was supported in part by the National Science Foundation under grants CNS-0845476 and CNS- 0834775

no inter-server communication load is incurred. Furthermore, replicating users' data on different servers also make the system less susceptible to bottleneck failures. However, data replication incurs a load of writing additional data to the database. Thus, there is a trade-off in terms of the communication load and writing load which in turn depends on the frequency of reading versus writing for different users.

Load Balance. Another important aspect affecting the system performance is to balance the loads among the servers, i.e., to prevent the scenarios where some servers are overloaded while others are idle. It is well-known that these scenarios can substantially degrade the performance of a distributed system.

Based on the discussion above, for a given user's communication pattern, the loads of the servers depend critically on the assignment of users to servers. The user communication patterns are defined by not only the social network of friends (typically represented as a relationship graph among friends), but also how often users log in Facebook (read), post messages (write), and the length of the posted messages. A good assignment will reduce the overall load and balance the individual loads among the servers. Furthermore, by replicating data across the servers judiciously, the overall load can also be further reduced.

II. Background

There are two main tasks of data management in OSNs: assignment users to servers and data replication. There are much work focusing on these such problems. In [1], author proposed a centralized algorithm for partitioning the network of user based on a relaxation of an integer program. The dynamical changes of the network as well as replication problem were not considered in this work. Algorithms proposed in [2] and are based on the community structure of the networks. Modularity is a very good parameter which represents the structure of the network as well as how users are related among themselves. However, computing modularity for the whole network is computationally expensive. Replication schemes proposed in [3] and [4] relies on a fixed partition scheme and a fixed number of replica is applied for all user data. When network structure changes, i.e nodes addition/deletion, a new assignment scheme might be needed to achieved before changing replication scheme would lead to better system performance. Moreover, social networks are very skew where most users having low degree of friend; and with a good assignment scheme the required replica is minimal and the redundant replica merely supports load balance among servers.

Our work focuses on distributed algorithm which adapts to the dynamical changes of the network and minimizes the global objective function value which is the sum of load incurred by users' activities. Unfortunately, this optimal assignment and data replication problem is NP-hard. Therefore, we introduce a distributed heuristic algorithm in which the servers perform local computations and exchange information among each other iteratively in such a way that the algorithm converges to a good assignment and replication in terms of (1) reducing the overall system load as well as (2) balancing the loads among the servers. In contrast with a centralized algorithm, a distributed algorithm offers the advantages of balancing the computations among all the servers and the ability to naturally adapt to the time-varying users communication patterns which is often the case for OSN applications.

III. Problem Formulation

We first define the following notations and derive the quantities of interest:

- N, M: Number of users and servers, respectively.
- V, S: A set of all users and servers, respectively.
- $N(v)$: set of users who has relationship with user v (v's neighbors).
- c_v^r, c_v^w, c_v^t: average cost of reading/writing and transferring v's data respectively. These costs are functions of v's total load amount.
- $P \in \{0,1\}^{N\times M}$: An assignment matrix where $p_{us} = 1$ if user u is assigned to a primary server s, and $p_{us} = 0$ otherwise. Since a user is assigned exactly to one primary server, $\sum_{s=1}^{M} p_{us} = 1$ and $0 \leq \sum_{u=1}^{N} p_{us} \leq N$, $\forall i$.
- $X \in \{0,1\}^{N\times M}$: A replication matrix where $x_{up} = 1$ if user u's data is replicated on server p, and $x_{us} = 0$ otherwise. Since user data is never replicated on its primary server, $x_{us} \leq p_{us}$.If user u's data is replicated on K servers, $\sum_{s=1}^{M} x_{us} = K$.
- r_v, w_v: the user v's average rates of reading and writing, respectively.

In this section, we first derive different types of load for a given user-server assignment without replication. These are necessary for the mathematical formulation and are the building blocks for the proposed distributed algorithms. Replication will be considered shortly. We now start with the read and communication load.

Read Load. User v's data is interested in by v itself and its friends. Then the total read load on a server s is:

$$L_s^r = \sum_{v\in V} p_{vs}\big(r_v c_v^r + \sum_{u\in N(v)} r_u c_v^r\big) \tag{1}$$

Communication Load. If v and u are assigned to the same primary server s, then there is no communication load. On the other hand, data need to be transferred between the two, and thus the total communication load of s is computed as:

$$L_s^c = \sum_{v\in V} p_{vs}\big(\sum_{u\in N(v)} r_u c_v^t(1-p_{us})\big). \tag{2}$$

where $p_{us} \in [0,1]$.

Write Load. Whenever a user v updates its data, a write load is incurred at its primary server, and total write load at server s is:

$$L_s^w = \sum_{v\in V} w_v c_v^w p_{vs}. \tag{3}$$

From all the above, the total load of all the servers without replication L^{wor} is computed as:

$$L^{wor} = \sum_{s \in S} L_s^r + L_s^c + L_s^w. \quad (4)$$

Data Replication.

Up until now we discuss the communication, read, and write loads assuming no data replication. Now if users' data is replicated on servers, as discussed previously, the communication load will be reduced. Therefore, the total change in the system load due to replication is:

$$L^{Rep} = \sum_{v \in V} \sum_{n \in S} x_{vn} \big(w_v c_v^w - c_v^t \sum_{u \in N(v)} r_u p_{un} \big) \quad (5)$$

The total load with replication L can be now rewritten as:

$$L = L^{wo} + L^{Rep}. \quad (6)$$

Let

$$\overline{L} = \tfrac{L}{M} \quad \text{and} \quad d_s = L_s - \overline{L}$$

denote the average server load and the imbalance load of server s, respectively. We define our objective function as:

$$F = L + \alpha \max_s d_s, \quad (7)$$

where $\alpha \in (, 1)$ is the weighted factor for controlling the trade-off between load imbalance and the total server load.

The problem of assigning users to servers and replication data is now casted as an integer programming problem with the optimization variables being p_{us} and x_{us}, denoting the primary user-server assignment and the replication scheme, respectively. Formally, the optimization problem is:

Minimize: F

Subject to:

$$\sum_s p_{us} = 1 \ \forall u \quad (8)$$

$$p_{us} + x_{us} \leq 1 \ \forall u \quad (9)$$

$$\sum_s x_{us} \leq K \quad (10)$$

$$p_{us} \in \{0, 1\} \quad (11)$$

$$x_{us} \in \{0, 1\} \quad (12)$$

The constraint (8) ensures that every user has exactly one primary server; constraint (9) guarantees that user u's data is never replicated on the same primary server; and constraint (10) restricts the maximum number of replica within system.

This optimization problem is unfortunately NP-hard. The proof is omitted due to limited space. Therefore, in the next sections we will present a distributed algorithm that produces an approximate solution.

Proposition 1: The global objective function is bounded by:

$$F_L \leq F < F_U \quad (13)$$

Where

$$F_L = \sum_{v \in V} \big(w_v c_v^w + r_v c_v^r + \sum_{u \in N(v)} r_u c_v^r \big) \quad (14)$$

$$F_U = F_L + \max \Big\{ L_{max}^c, L_{max}^{Rep} \Big\} + \alpha \frac{N-1}{N} F_L \quad (15)$$

with

$$L_{max}^c = \sum_{v \in V} \sum_{u \in N(v)} r_u c_v^t (1 - p_{us})$$

and

$$L_{max}^{Rep} = \sum_{v \in V} w_v c_v^w \min\{|N(v)|, N\}$$

Proof: The detailed proof is omitted due to the lack of space. ■

IV. Distributed algorithm/protocol

In this section, we propose an approximate distributed algorithm for solving the optimization problem above. The algorithm consists of two parts. The first part is a greedy method to provide quick convergence to a local solution. The next part employs simulated annealing method to overcome the local solutions. For the first part, the main idea is: at each time step, the most overloaded server is selected to (1) reassign some of its users to other servers or (2) replicate its user data to other servers in order to reduce its load. By appropriate taking either one or two actions above, both the total load and and load imbalance will be simultaneously reduced over time. For such an algorithm to work, some small amount of information must be kept and exchanged among the servers to allow the servers to know immediately whether (1) it is the most overloaded server and (2) whether it should take actions (1) or (2). The second part, simulated annealing methods will also be computed locally. In what follows, we show the following information and local computations are sufficient to perform such a distributed algorithm.

Local information and computation. Each server is assumed to handle a number of users, and thus it maintains all information pertained to its local users. Specifically, this information includes a table of total read load incurred by each users due to its friends from other users. Let us define the following locally stored information at server s:

- N^s, V^s: The number and the set of users primarily assigned to server s, respectively.
- $L^s_{N^s \times (M-1)}$: A matrix represents the average load due to v's data requests from other servers whose entries $L^s(v, n), n \in S \setminus s$ denotes the load of reading and transferring v's data to its friends located on server n. Recall that $L^s(v, n) = \sum_{u \in N(v)} p_{ut}(1 - x_{vn}) r_u c_v^r$.
- N^{rs}, V^{rs}: : The number and the set of users whose data are replicated on server s.
- $k(v)$: number of v's replica.

Given the information above, the total read and communication loads at server s can be computed locally as:

$$L_s^r = \sum_{v \in V^s} \big(r_v c_v^r + \sum_n L^s(v, n) \big) \quad (16)$$

Similarly, the total write load at server s can be computed locally as:

$$L_s^w = \sum_{v \in V^s} w_v c_v^w + \sum_{u \in V^{rs}} w_u c_u^w \quad (17)$$

The total load that server s contributes to the overall system load is:

$$L_s = L_s^r + L_s^w \quad (18)$$

Now, each server exchanges their total loads among each other. Thus, every server will be able to determine whether it is the maximum load server. Next, given that a server is the maximum load server, it will either (1) reassign some of its users to other servers or (2) replicate its user data to other servers in order to reduce its load. To choose either of the actions, the maximum load server will use a greedy approach that takes the action out of two that result in the maximum reduction of the objective value F. The reduction of the objective value due to each of the actions can be estimated locally as follows.

Load Change due to Replication. If user v's current data on s, is replicated on server n, then the local read and communication load at server s will reduce by $L^s(v,n)$ since $L^s(v,n) = 0$ after the replication action. The write load at server t will increase by w_r. The load at server s will decrease by $L^s(v,n)$, the load at server n will increase by $w_v c_v^w$ and the overall system load will change by:

$$\Delta L = w_v c_v^w - L^s(v,d) \quad (19)$$

Load Change due to Reassignment of Primary Server. Assume s is the primary server of v, then by reassigning user v to another server n, the load at server s will decrease by $L^s(v,n) + w^v c_v^w$, the load at server n will increase by $\sum_{u \in N(v)} p_{us} r_u c_v^r + (1 - x_{vn}) w^v c_v^w$, and the change in the overall system load is:

$$\Delta L = \sum_{u \in N(v)} p_{us} r_u c_v^r - L^s(v,n) - x_{vn} w^v c^w \quad (20)$$

Load Change due to Replication Removal. Often, due to greedy move, it is possible that one gets a lower objective value by backtracking via removing a replication. Therefore, we will calculate the reduced load due to removing a replication. If we remove u's replication on server s, the load at server s will increase by $\sum_{v \in N(u)} p_{vs} r_v c_u^t - w^u c_u^w$, and the change in overall system load is:

$$\Delta L = \sum_{v \in N(u)} p_{vs} r_v c_u^t - w^u c_u^w \quad (21)$$

For each of the change in load, we will also be able to compute the change in F which include the load imbalance since the loads from all other servers are available.

It is important to note that (19),(20),(21) can be computed locally.

Thus, we propose the following distributed algorithm (DAROS) for user-server assignment and replication. There are two stages in DAROS: optimization mode and convergence mode. In optimization mode, only overload servers involve in the optimization process. Each overload server selects each currently assigned user and computes the gain of reassigning user to a new server or replicating user's data only and/or removing current replication. The decisions are made based on which action will result in better gain. The process of replicating and reassigning user in the optimization mode will be iterated until no more change in user-assignment is observed, i.e., the algorithm probably converges to a local solution. We call this the convergence mode. To overcome the local solution, in the convergence mode, all servers will compute the gain of each tentative action (reassign, replicate or remove replication) and make a decision with probability of $\min\{1, exp(-\Delta F/T_b)\}$ where ΔF is the estimated gain and T_b is an experimental coefficient for controlling the convergence rate of the overall DAROS. We used T_B as a function of global objective value in such a way that if the global objective function's value is near the global optimal point, the probability of perturbation is low, i.e $T_b = \beta \frac{F - F_L}{F_U - F_L}$. This is the basis of the simulated annealing method. We now describe with the first stage of the algorithm.

TABLE I: Simulation results

(M,K)	% Global cost saved	%Inter-server	% Repl. cost
(8,1)	46.22	8.14	4.61
(8,3)	46.35	4.37	7.87
(8,7)	46.70	3.42	8.21
(16,1)	46.54	15.06	3.7
(16,3)	46.63	11.42	6.88
(16,7)	43.85	12.86	7.59

Algorithm 1 DAROS

```
Input: {V^s}, {L^s}, K
Output: p_us, x_us
Assign equal number of users to each server uniformly at random.
Set mode to optimization
repeat
    if mode is optimization then
        Run algorithm 2
        if No action has been made then
            Switch to convegence mode
        end if
    else//Convergence mode
        Run algorithm 3
        Switch to Optimization mode
    end if
until Freeze
```

V. Simulation Result

In this section, we show our preliminary simulation results. In particular, we use the Facebook data [5] which has N = 63731 users and 817,090 relationships. Simulation parame-

Algorithm 2 Optimization mode

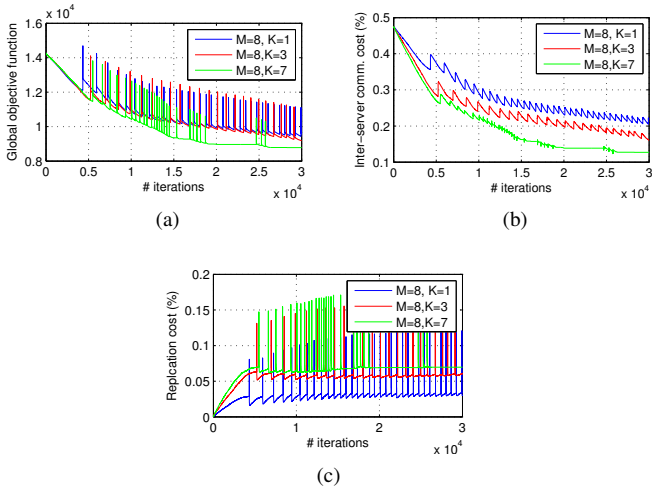
At each time step, the servers exchange their loads among each other.
if s is the most overloaded server **then**
  for each v in V^s **do**
    Reassign users based on $\Delta F_{Reassign}$ (20)
    Replicate user data based on ΔF_{Repl} (19)
  end for
  % Now we perform backtrack since during the previous greedy move might be suboptimal.
  for each v in V^{rs} **do**
    Let $F\_removing$ = change in objective value by removing v's replication using (21)
    if $F_{removing} < 0$ **then**
      Removing v's replication on s
      Decrease $k(v)$
    end if
  end for
end if

Algorithm 3 Convergence mode

Set $T_B = \beta \frac{F - F_L}{F_U - F_L}$
for each v at every server **do**
  for each $n \in S \setminus s$ **do**
    Compute $F_{Reassign}$ by (20)
    Reassign v to t with
      probability $\min\{1, exp(-\frac{\Delta F_{Reassign}}{T_b})\}$
    if There is no v's replication on t **then**
      ΔF_{Repl} by (19)
      Replicate v's data on t
        with probability $\min\{1, exp(-\frac{\Delta F_{Repl}}{T_b})\}$
    end if
    for each v in V^{rs} **do**
      ΔF_{Remove} by (21)
      Remove v's data replication s
        with probability $\min\{1, exp(-\frac{\Delta F_{Remove}}{T_b})\}$
    end for
  end for
end for

ters are: ReadRate=1, WriteRate=0.1, ReadCost=0.01, WriteCost=1, TransferCost=0.02, $\alpha = 0.5$; (M,K) = (8,1),(8,3),(8,7).

Fig. 1(a), Fig. 1(b), and 1(c) show the changes in global objective function's value, the inter-server communication cost, and the replication cost, respectively. Initially, the servers are assigned an equal number of users uniformly at random. Thus, a simple algorithm of assigning equal number of users to each server would have the performance similar to that of the proposed algorithm at $iteration = 0$. As seen, the objective function decreases significantly over time, showing the significant benefit of our proposed algorithm over the naive method. Each spike in the graph corresponds to the simulated annealing method to overcome the local minimums. Once the users are perturbed in convergence mode, the communication and replication cost increase dramatically. However, at next iterations, these costs reduce significantly due to appropriate reassignment and replication. As seen, this helps the algorithm to further reduce the objective value and inter-server communication cost. The simulation results (table I) show that the global cost is saved by about 46% compared to the random assignment/replication scheme. The inter-server communication cost and the replication cost are just small fractions of the global cost.



(a) (b) (c)

Fig. 1: (a) Objective function values b) Inter-server communication cost (%) c) Replication cost (%).

VI. Conclusion

We study the problem of assigning users to servers and data replication in a distributed manner for online social networking (OSN) applications. This problem is NP-hard. Therefore, we introduce a distributed heuristic algorithm that finds a good assignment and replication in terms of reducing the overall system load as well as balancing the loads among the servers. In contrast with a centralized algorithm, a distributed algorithm offers the advantage of balancing the computations among all the servers and the ability to naturally adapt to time-varying user's communication patterns. Simulations results show the effectiveness of our proposed algorithm.

References

[1] H. Nishida and T. Nguyen, "Optimal client-server assignment for internet distributed systems," in *ICCCN 2011*.

[2] M. Newman, "Modularity and community structure in networks." in *Proc Natl Acad Sci USA (2006)*.

[3] D. A. Tran, K. Nguyen, and C. Pham, "S-CLONE: Socially-aware data replication for social networks," *Comput. Netw.*, vol. 56, no. 7, pp. 2001–2013, May 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.02.010

[4] K. Nguyen, C. Pham, D. A. Tran, and F. Zhang, "Preserving social locality in data replication for social networks," in *Proceedings of IEEE ICDCS 2011 Workshop on Simplifying Complex Networks for Practitioners (SIMPLEX 2011)*, Minneapolis, MN, June 2011.

[5] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in Facebook," in *Proc. Workshop on Online Social Networks*, 2009, pp. 37–42.