9th IEEE International Conference on Cloud Computing (IEEE CLOUD 2016), June 27 - July 2, 2016, San Francisco, USA, IEEE Press

Improving Cloud-based Online Social Network Data Placement and Replication

Hourieh Khalajzadeh[*], Dong Yuan[+], John Grundy[!], Yun Yang[*]
[*]School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia
[+]School of Electrical and Information Engineering, the University of Sydney, Sydney, Australia
[!]School of Information Technology, Deakin University, Melbourne, Australia
[*]{hkhalajzadeh, yyang}@swin.edu.au
[+]dong.yuan@sydney.edu.au
[!]j.grundy@deakin.edu.au

***Abstract*—Online social networks make it easier for people to find and communicate with other people based on shared interests, values, membership in particular groups, etc. Common social networks such as Facebook and Twitter have hundreds of millions or even billions of users scattered all around the world sharing interconnected data. Users demand low latency access to not only their own data but also their friends' data, often very large, e.g. videos, pictures etc. However, social network service providers have a limited monetary capital to store every piece of data everywhere to minimise users' data access latency. Geo-distributed cloud services with virtually unlimited capabilities are suitable for large scale social networks data storage in different geographical locations. Key problems including how to optimally store and replicate these huge datasets and how to distribute the requests to different datacenters are addressed in this paper. A novel genetic algorithm-based approach is used to find a near-optimal number of replicas for every user's data and a near-optimal placement of replicas to minimise monetary cost while satisfying latency requirements for all users. Experiments on a large Facebook dataset demonstrate our technique's effectiveness in outperforming other representative placement and replication strategies.**

Keywords-Online Social Network; Data Placement; Data Replication; Latency; Genetic Algorithm;

I. INTRODUCTION

Online social networks usually have very large numbers of users geographically distributed all around the world sharing different types of data, some like videos and images very large, with each other. The size and number of such data items are growing dramatically every day. Furthermore, these users have specific expectations including low latency, data consistency and availability, and privacy requirements from their social network service provider. Users can tolerate a certain threshold to access their own data or their friends' data. However, not being able to access this data in the desired time is likely to lead users to become disappointed about the online social network, lowering their usage and thus advertising and other provider revenue. A possible solution would be to store the data related to every user in every available datacentre. However, as different copies of user data may need updating regularly, and due to its very large size, such a huge investment becomes infeasible and uneconomic. Hence, there is always a trade-off between the data storage cost and latency.

Nowadays, many social network providers use their own private datacentres to store users' data. However, building private datacentres is extremely expensive and it is normally not an option for every social network provider. Even large providers are concerned about the rapidly increasing storage, data transmission and datacentre energy usage and monetary costs. To reduce the cost related to datacentre setup and maintenance, a better solution is to make use of cloud datacentres. Cloud computing is a technology trend where users can rent software, hardware, and infrastructure on a per use (compute and/or data) basis. However, as many privacy issues exist in using cloud datacentres and social network providers have to trust cloud providers to share their data with, combining private and public cloud datacentres could be a smart solution. In addition, cloud rental is also very costly and energy expensive if naïve social media data replication and distribution were used.

There are many cloud providers with different datacentres around the world that facilitate the setting up, managing, and maintaining private storage infrastructure. Amazon S3, Google Cloud storage, and Microsoft Azure are some examples. By using cloud datacentres, social media service providers could store users' data in every geographical location to satisfy the latency requirement for users with much lower cost. However, as use of any resource needs to be paid for, the cost for storing data and updating data would be still huge if they store the users' data in all datacentres. Hence, they need to store users' data in such a way for all users to access data in a tolerable time while having a minimised cost.

For example, let us assume that we have two users, one in Singapore and the other in California, sharing data with each other. One solution could be storing and replicating their data in both S3's North California datacentre and Singapore datacentre, and pay for the storage cost in both datacentres. Another solution could be storing some or all

data in just one of these datacentres to reduce the storage cost. However, by doing so, one of the users has to suffer a higher latency. A more appropriate solution could be to store their data in a datacentre in between, which has relatively low latency to both users, such as S3's Tokyo datacentre. Thus, both users can have a tolerable latency by paying only one time storage cost. Hence, we need to explore all possible placement strategies to find out the best one.

This data placement and replication challenge is currently unsolved. The placement and replication issue involves finding not only the best place to store the data, but also the suitable number of replicas to ensure the latency requirement for all users to access their own data and their friends' data. Taking the latency for users and all friends to access the main user's data into account makes our work different from others. To minimise the monetary cost while guaranteeing latency requirement, we used a Genetic Algorithm (GA) to find the most suitable number of replicas and their placement for every user. Our GA is able to find solutions which could not be found by even complex rational strategies as it explores different random placements in order to find the best one. Our goal is to find a replication of a given set of users' data with minimum storage cost while guaranteeing that p^{th} percentile of latencies is less than the desirable latency, i.e. over p% of all operations are within the specified latency requirement. The SNAP Facebook dataset [1] is used to test our prototype and experimental results reveal the effectiveness of our algorithm. As verified in simulation experiments, our GA-based data placement and replication strategy is capable of finding good solutions in most cases.

The remainder of this paper is organised as follows. Section 2 gives a motivating example of online social network data storage and analyses the research problem. Section 3 introduces the system model and the cost model of online social networks. Section 4 presents the detailed genetic algorithm used in this paper. Section 5 demonstrates the simulation results and the evaluation. Section 6 discusses related work. Finally, Section 7 addresses our conclusions and future work.

II. Motivating Example and Problem Analysis

A. Motivating example

Online social networks deal with a very large scale of users distributed all around the world sharing a growing volume of interconnected, increasingly large data. Users typically have friends in diverse places who expect to access their data in a tolerable time. For instance, Facebook as the world largest social network passes 1.55 billion monthly active users and 1.01 billion daily active users in 2015 [2]. Based on a research in 2012 [3], 500+ Terabytes of data are ingested to Facebook every day which is for almost 550 million daily active users out of 950 million users in 2012.

Facebook announced an investment of more than $1 billion in the infrastructure that powers its social network, which serves more than 845 million users a month around the globe in 2012. The company spent $606 million on servers, storage, network gear and datacentres in 2011 and another $500 million in 2012 [4]. Due to the growing number of users and their data size, Facebook has more than 10 private datacentres in 2015 and it stated that it still needs to extend its datacentres in the future to fulfil the expectations of all users. As described above, it is obvious that data storage in social network applications is a data intensive job and every one cannot afford setting up many datacentres in different locations all around the world. Many social network providers such as Dropbox [5] are using cloud datacentres as a more affordable solution to store their data.

B. Problem analysis

Traditionally, due to the lack of Internet based computing systems such as cloud computing, service providers had to set up and maintain their own datacentres. With the advent of cloud computing [6], online social network providers can benefit from storage in cloud datacentres. There are some advantages to using cloud datacentres such as being ready-to-use, scalable, cost beneficial, reliable, and manageable. Furthermore, they can use geo-distributed datacentres all over the world without any extra investment to fulfil the latency requirement for users distributed geographically all around the world.

However, there are also some disadvantages, such as security and privacy, limited control and flexibility, technical difficulties and downtime, and vendor lock-in issues [7] in the cloud which lead some providers to combine using private and public datacentres as a hybrid infrastructure.

When using both private datacentres and geo-distributed cloud datacentres to store social media data, every user needs to have a primary copy of data and several secondary replicas to ensure the latency requirement for his/her friends who want to access his/her data. The issue that service providers have to address is to find the most appropriate number of replicas for every user's data and their locations by finding the trade-off between monetary cost and latency. Hence, we need an algorithm to find the minimum cost storage strategy for data placement and replication in cloud while guaranteeing service level agreement such as latency for all users. This is currently an unsolved problem.

III. Problem Formulation and Cost Model

The research problem addressed in this paper is data placement and geo-replication of online social network services while optimising service provider's monetary expenses in using resources of geo-distributed clouds and guaranteeing service level agreements such as latency for service users. We do not include the data transfer cost because data needs to be transferred to the users regardless of where they are located, i.e. no extra data transfer cost is involved and it is reflected in latency. The data update cost is not considered here because our system is static and handling updating of data is postponed to the future.

Every user has a primary copy located in their primary datacentre, which is the nearest datacentre to their location. It is assumed that all users read their own data from their primary datacentre and every friend of them reads their data from their nearest datacentre which stores any secondary data of their data. It is also assumed that every write

operation goes to the primary datacentre. There are M datacentres and N users, each with one dataset.

The users and their collection of datasets stored in different datacentres are denoted respectively as:

$$U = \{u_1, u_2, ..., u_N\}$$

$$D = \{d_1, d_2, ..., d_N\}$$

Datacentres in the system are denoted as:

$$S = \{s_1, s_2, ..., s_M\}$$

The solution space is a matrix X of size $N \times M$ as follows:

$$x_{ij} = \begin{cases} 1 & \text{Data of user } i \text{ is stored in datacentre } j \\ 0 & \text{Otherwise} \end{cases}$$

A. Cost Model

"Cost" as used in this work is the cost for storing data in different datacentres. Considering N as the number of users, and R_i as the number of replicas for user i, cost is the total monetary cost of storing main copy and replicas of all users' data in different datacentres for a specific duration and is calculated as follows:

$$Cost(\$) = \sum_{i=1}^{N} StorageCost_i \quad (1)$$

where

$$StorageCost_i = UnitStoragePrice \times StoredDataSize_i \times (R_i + 1)$$

The *UnitStoragePrice* is the price for storing one Gigabyte of data per month in a datacentre and $StoredDataSize_i$ is the data size for user i. Thus, the storage cost is the cost for storing user's data and replicas for one month in different datacentres.

B. Latency

Latency between users and datacentres is calculated using an approximation based on distance. Every user has a primary datacentre that is the nearest datacentre to their location. We assume that every user has a latency of 20 *ms* with their primary datacentre and the latencies between the user and other datacentres are calculated based on (2) [8]:

$$Latency\ (ms) = \begin{cases} 20 & \text{User and datacentre are in same region} \\ 0.02 \times Distance(km) + 5 & \text{Otherwise} \end{cases} \quad (2)$$

Every user reads data from the nearest datacentre that has a copy of the data. Thus, the final latency for every user is the summation of the latency between them and their data and the latency between all their friends and the nearest secondary replicas to them. The targeted maximal average response delay per request is set to 150 ms and 200 ms, since latency more than 200 ms will deteriorate the user experience significantly [8]. We can use alternative default latency to local datacentre and alternative coefficients for remote datacentres. We could also include time-of-day and other refinements that impact both latency and cost.

C. Problem formulation

We aim to minimise the cost while satisfying service level agreements, in our case primarily maximum permitted latency. We can also include other factors such as energy consumption (watts to store/retrieve/transmit), and reliability (probably retrieve/transmit fails). The problem using desired latency is formulated as follows:

minimise:

$$Cost = S$$

where

$$S = \sum_{i=1}^{N} StorageCost_i$$

is the cost for storing main data and its replicas, subject to:

$$\sum_{i=1}^{N} latency_i \le DesiredLatency \quad (3)$$

This constraint means that the latency for every user must be lower than the desired latency in order to ensure the latency requirement for every user. The latency is the latency for user i and all his/her friends to access his/her data. For every user i, we have the following constraints.

$$\sum_{j=1}^{M} p_{ij} = 1 \qquad \forall i \in U \quad (4)$$

$$p_{ij} + s_{ij} \le 1 \qquad \forall i \in U, \forall j \in S \quad (5)$$

$$\sum_{j=1}^{M} s_{ij} \ge R_i \qquad \forall i \in U \quad (6)$$

In these constraints, p_{ij} and s_{ij} indicate existing primary and secondary replicas of user i's data in datacentre j. Constraint (4) ensures every user has a single primary replica in all datacentres. Constraint (5) ensures that no primary and secondary replicas of the same user are co-located in a common datacentre. Finally, constraint (6) specifies the minimum number of secondary replicas R_i for every user i to ensure the data availability.

IV. A Genetic Algorithm Based Placement Strategy

The data placement and replication problem defined in the previous section has many decision variables due to the large number of social network users. A GA-based placement and replication strategy is proposed to find the most cost effective number of replicas for users' data and their placement while guaranteeing latency requirement defined in service level agreement.

GA is a search technique often employed to find the exact or approximate solutions for optimisation and search problems. GA is a specific class of evolutionary algorithms inspired by evolutionary biology. In GA, every solution is represented with a string, also known as a chromosome,

which follows the semantics defined by the encoding method. After encoding, the candidate solutions, i.e., the initial population, need to be generated as the basic search space. Within each generation, three basic GA operations, i.e., selection, crossover, and mutation, are conducted to imitate the process of evolution in nature. Finally, after the stopping condition is met, the chromosome with the best fitness value is returned, representing the best solution found in the search space. This ends the GA process [9].

An overview of our GA-based social media data placement and replication strategy is presented in Fig. 1. We have the social graph of users and their connections, latency related to different datacentres, and the desired latency requirement to calculate and compare the latency. Additionally, users' data size and the storage cost are used to determine cost. Latency and cost are calculated using the fitness function. To avoid violating latency by GA operations, after every crossover and mutation, latency is being checked. Primary data cannot be mutated as the main data have to be stored in the user's main datacentre, the closest to their location.

Genetic algorithm for social network data placement and replication

Input:
Rate of crossover: r_c
Rate of mutation: r_m
Size of population: *popsize*
Size of selected population: *keep*
Number of iterations: *epoch*
Output:
Solution: X

// **Initialisation**
1 generate *popsize* feasible solutions randomly;
2 save them in the population *pop*;
// **Loop until the terminal condition**
3 for i=1 to *epoch* do
// **Crossover**
4 for j=1 to *popsize-1* do
5 randomly select two solutions x_a and x_b from pop;
6 generate x_c and x_d by two-point crossover from x_a and x_b under rate r_c
7 if latency requirement is valid, save x_c and x_d to *pool*;
8 update *newpop* = *pop* + *pool*;
9 endfor
// **Mutation**
10 *Len* = size of *newpop*
11 for j=1 to *Len* do
12 select a solution x_j from *newpop*
13 mutate each bit of x_j under rate r_m and generate a new solution x'_j
14 if latency requirement is valid, update x_j with x'_j in *newpop*;
15 endfor
16 endfor
// **Selection**
17 using tournament selection, select *keep* solutions from *newpop* and save them in *pop*;
// **Returning the best solution**
18 return the best solution x in *pop*;

Figure 1. Pseudocode of the GA algorithm

A. Initial population generation

The strategy starts with the encoding of the users' data replicas placement in different datacentres. Here, as depicted in Fig. 2, what we have employed is a two-dimensional encoding where the first dimension denotes users' ID as an indicator of users' data and the second dimension denotes the ID of different datacentres. Matrix x_{ij} is initialised with random 1s and 0s showing whether user i's data is stored in datacentre j or not respectively.

Users \ DCs	1	2	3	...	D
1	0	0	1	0	1
2	0	1	1	1	1
3	1	1	1	0	0
...					
N	0	0	0	1	0

Figure 2. Problem encoding

The fitness function is considered as the cost of storing data replicas of all users in different datacentres. Hence, the fitness function is calculated as follows:

$$F(x) = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} \times UnitStoragePrice \times StoredDataSize_i$$

A validation process where generated chromosomes are checked with desired latency is done during this step. The latency requirement is checked and the valid chromosomes are then retained and the invalid ones are discarded and replaced with newly generated ones.

The first genetic operation is selection, where tournament selection is used, which involves running several tournaments among a few chromosomes chosen at random from the population and the winner of each tournament is selected. The reason of using tournament selection is that it prevents too quick convergence as rank selection while it is computationally more efficient, as there is no need to sort the whole population which is a potentially time consuming procedure [9].

B. Crossover procedure

The basic idea of the GA crossover operation is that a random crossover point is chosen first and then the segments of parents are swapped at that point to produce new children. Therefore, children inherit the characteristics of both parents. For two random chromosomes, a two point crossover is conducted with a specific probability of 80%. An example of the crossover process is presented in Fig. 3.

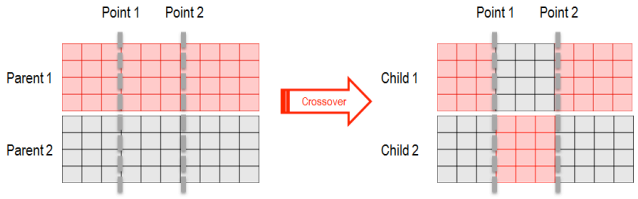

Figure 3. Two point crossover used in our method

C. Mutation procedure

In GA-based mutation, which is depicted in Fig. 4, the stored user's replica is mutated at a randomly selected cell of a chromosome. The mutation rate is set to a small probability value such as 10% since mutation can easily destroy the correct topological order and result in invalid solutions.

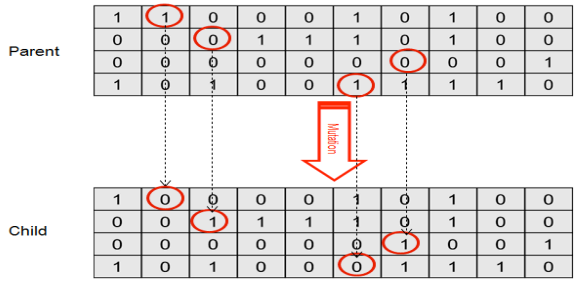

Figure 4. Mutation used in our method

At the end of each generation, the chromosomes with the best fitness values of each generation are chosen and the children with the worst fitness value are removed from the considered population. The genetic evolution process repeats itself until the stopping condition is satisfied. Finally, the best solution is returned.

V. Simulation Results and the Evaluation

Our new GA-based data placement and replication strategy is generic and can be used in any social network application fitting our data placement approach and social relationships graph. In this section, we demonstrate the simulation results and comparison of our benchmark with different placement and replication strategies. The SNAP (Stanford Network Analysis Project) real world Facebook dataset [1] was used to demonstrate how our algorithm finds an efficient data placement and replication with the minimised cost while satisfying the latency requirement.

A. Experiment dataset and setting

SNAP is an undirected Facebook dataset with 4,039 users and 88,234 relationships which is used in the experiments. This dataset contains a social graph of users IDs and the relations between them. Facebook data was collected from survey participants using their Facebook app. Two types of experiments were conducted: Section B evaluates the cost reduction of GA per iteration and its effectiveness while Section C shows the efficiency of our strategy comparing with other strategies.

As we did not have the users' information such as location in the introduced dataset, we generated random locations in the US for users based on their latitude and longitude. Moreover, 10 datacentres are assumed in the real locations of Facebook datacentres in Oregon, North Carolina, Altoona, Silicon Valley, Santa Clara, San Jose, San Francisco, Ashburn, Virginia, and Council Bluffs [10]. The nearest datacentre is chosen for every user as the primary datacentre. Number of users around each datacentre who choose this datacentre as their primary datacentre is shown in Fig. 5. The unit storage cost for data storage in all datacentres is considered as $0.125 per GB per month. This could be refined to use different values per datacentre if desired.

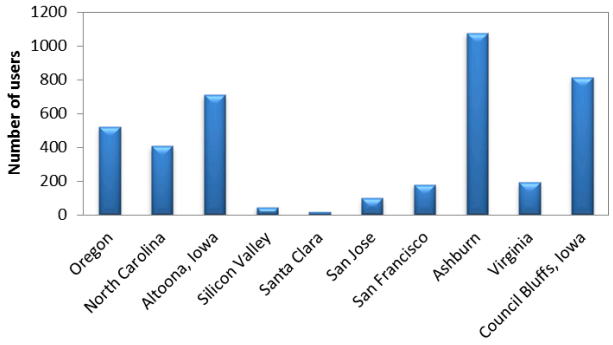

Figure 5. Number of users located around different datacentres

Based on the information explained in Section 2 that Facebook is collecting 500 terabytes of people's data every day and due to the 950 million population of Facebook and 550 million daily active users in 2012 when this dataset was collected, on average, every active user stores 900 KB (500 TB / 550 Million) information daily in a Facebook datacentre which is the amount of 27 MB (900×30) monthly. This data size increases every month. We generated random sizes of data for users following a normal distribution with this average size as the mean.

B. Evaluation of cost effectiveness

To further explain the GA setting, chromosomes are considered as a matrix of $N \times M$ with N as the number of users and M as the number of datacentres. N is 4039 and M is 10 in our experiments. Population size is considered as 30. Crossover with crossover rate of 0.8 and mutation by mutation rate of 0.1 are considered [11]. Selection is based on the tournament selection. In each iteration, half of the best parents and newly generated children are kept for the next iteration. Fitness function is considered as the cost of every solution as described before. Latency requirement is considered as a constraint and solutions which do not meet it had been removed.

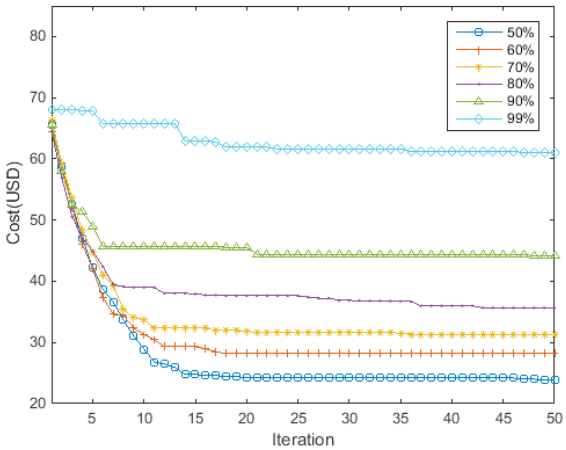

Figure 6. Cost reduction per iteration using the genetic algorithm for different percentiles of a desired latency of 150ms

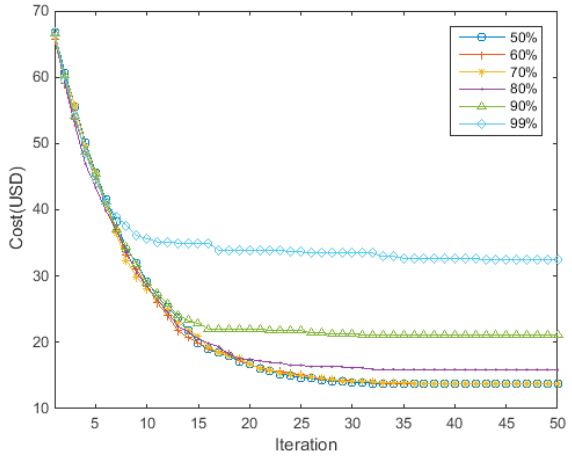

Figure 7. Cost reduction per iteration using the genetic algorithm for different percentiles of a desired latency of 200ms

The termination condition is based on the number of iterations. 50 iterations were used as no more cost reduction was observed after 50 iterations. The cost reduction per iteration with a different percentile (50%-99%) of latencies fulfilled (150 and 200 ms) is depicted in Figs. 6-7. For instance, in Fig. 7, the green line shows the cost reduction from the first iteration of GA data placement until the final placement while 90% of users have the latency less than 200 ms for themselves and their friends to access their own data.

As an example, referring to Fig. 7 with latency requirement of 90 percentile of latencies less than 200 ms, by considering the user size for all 4039 users and the unit storage cost as described previously, the initial cost resulted by the first iteration of GA is $66.633 with average number of replicas as 5. The minimum cost found by GA in the 50th iteration is $21.147 with an average replica number of 2. Moreover, the 90th percentile latencies for these two placements are 120.7639 ms and 199.9593 ms respectively which are both acceptable, based on the latency requirement of 200 ms. Thus, the cost reduction for 4039 users with average data size of 27 MB is $45.486. Time for running 50 iterations is 705.5696 minutes. We used a general purpose EC2 instance with vCPU=2, ECU=6.5, and Memory (GB) = 8 for our simulations which costed $0.177 per hour. Thus, 705.5696 minutes, i.e. 11.75 hours, for running GA costs around $2. Hence, the total cost reduction of $45.486 minus the EC2 instance cost of $2.079 would be $43.407 for 4039 users. This means the cost reduction percentage of around 65% which could thus be millions of dollars per month for a social network application with the user size of Facebook.

C. *Evaluation of different strategies*

Different strategies to replicate and place the described Facebook users' data in different datacentres which were simulated and compared with our strategy are as follows:

- The first strategy is our GA-based algorithm in which one copy of data is stored in the nearest datacentre. Genetic algorithm is used to find the optimised number of replicas and the best placement for them.
- Random placement and replication of data in different datacentres. The minimum number of replicas is 1 because we should have one main copy of data and the maximum is 10 as we have 10 datacentres.
- Placing one copy of data in a random datacentre.
- Placing two copies of data in two random datacentres.
- Placing three copies of data in three random datacentres.
- Full replication of every data in all datacentres.

Datacentres are sorted based on the distance for every user in the next 3 strategies. Because long distance causes high latency, every user prefers to have a copy of data in his/her nearest datacentre.

- One copy of data is stored in the most preferred datacentre of every user.
- Two copies of data are stored in the first and second preferred datacentres.
- Three copies of data are stored in the three most preferred datacentres.

Datacentres are sorted based on both distance as list1 and number of friends as list2 for every user in the next two strategies.

- One copy of data is stored in the most preferred datacentre in list1 and one more copy is stored in the most preferred datacentre in list2.
- One copy of data is stored in the most preferred datacentre in list1 and two more copies are stored in the two most preferred datacentres in list2.

Different settings are assumed to compare the results of these strategies. These settings are based on the service level agreements on the latency requirement for users and their friends to access their data. Latency requirement is defined as: "pth percentile latency must be lower than the desired latency" which means that over p percent of the latencies are less than the desirable latency. Requirements are assumed as 50%, 60%, 70%, 80%, 90%, and 99% of the latencies are less than 150 ms and 200 ms.

Based on subsection B, no more significant cost reduction was seen after 20 iterations, for the purpose of time efficiency. To repeat the experiments five times and compare the average results 20 iterations were used for GA in this step. As the percentage more than 90% makes much more sense in a most of the applications [12], the results for 99.99% latencies lower than 200 ms are depicted in the Fig. 8. We used 99.99% to ensure that nearly all of the users can access their own data and all their friends in the desirable latency.

As shown in Fig 7, the only strategy, except costly full replication, that can guarantee the latency requirement of "99.99% latencies lower than 200 ms" with a reasonable cost is GA which shows the outstanding performance of our strategy comparing with other strategies. Therefore, our GA based strategy can find the minimised cost while guaranteeing the latency requirement for nearly all users.

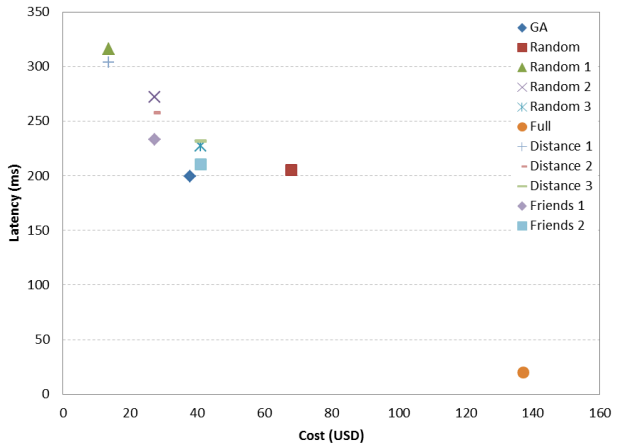

Figure 8. Comparison of different strategies with latency requirement of 99.99% lower than 200 ms

VI. Related Work

Many papers in the literature focus on energy efficient workload placement, virtual machine placement, applications scheduling, load balancing, task scheduling, resource allocation in the cloud, and job scheduling and data replication in the grid. These are not comparable with our work as we focus on data placement and replication in the cloud. Therefore, in this section, we compare our work with existing literature in three categories: first, optimising online social networks services, second, use of evolutionary algorithms for data placement and replication, and third, data placement and replication in cloud.

Optimising online social networks (OSNs): For OSN at a single site with different servers, social locality is maintained to address this issue in literature. SPAR [13] minimises the total number of slave replicas while maintaining social locality for every user; S-CLONE [14] maximises 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 datacentres to reduce the total inter-data-centre traffic, and others propose a framework that captures and optimises multiple dimensions of the OSN system objectives simultaneously [15]. Other works do not involve QoS as in our geo-distribution case.

Using of evolutionary algorithms for data placement and replication: To decrease the network traffic and undesired long delays in large distributed systems such as Internet, replicating some of the objects at multiple sites is considered as one possible solution in [16]. The decision of what and where to replicate is solved by genetic algorithms. Normal GA is considered for static situations and a hybrid GA is proposed that takes current replica distribution as input and then computes a new one using knowledge about the network attributes and the changes occurred. Furthermore, problem of co-scheduling job dispatching and data replication in wide-area distributed systems in an integrated manner is addressed in [17]. Their system contains three variables as the order of the jobs, the assignment of the jobs to the individual compute nodes, and the assignment of the data objects to the local data stores. A genetic algorithm is used to find the optimal placement. However, these do not consider the social network data placement problem in the cloud.

Some data placement strategies based on genetic algorithms are proposed in [18] and [19] to reduce data scheduling between cloud datacentres and the distributed transaction costs as much as possible. Additionally, the problem of placing the components of a SaaS and their related data in the cloud is addresses in [20]. However, data replication is not considered in these papers.

Data placement and replication in the cloud: The inter-datacentre communication of the online social network services is focused in [21]. Moreover, a geo-cloud based dynamic replica creation in large global Web sites such as Facebook is presented in [22]. Volley [23] addresses the automated data placement challenge which deals with WAN bandwidth costs and datacentre capacity limitations while minimising user-perceived latency. Additionally, the cloud storage reconfiguration while respecting application-defined constraints to adapt to changes in users' locations or request rates is addressed in [24]. However, they do not consider the monetary cost for replicating data in their work.

A mechanism for selectively replicating large databases globally is introduced in [25] to minimise bandwidth. However, they replicate all records in all locations either as a full copy or as a stub. Using geo-distributed clouds for scaling the social media streaming service is advocated in [8] to address the challenges for storing and migrating media data for timely response and moderate expense. They work on videos and focus on resource and data migration. The primary focus in [26] is to minimise the cost incurred by latency-sensitive application providers while satisfying consistency and fault-tolerance requirements with taking workload properties into account. However, latency definition in their work makes it not comparable with our work.

The monetary expense of the OSN service with considering its QoS, data availability requirements, inter-cloud traffic as well as the carbon footprint of OSN services is investigated in [15]. The social locality assumption in which they have to keep all friends' replica in one's main datacentre makes their work not comparable with ours. Multi-objective optimisation including reducing the usage of cloud resources, providing good service quality to users, and minimising the carbon footprint is studied in [27]. However, they consider latency as an objective instead of constraint which makes it not comparable with our work.

Placing and replicating the data related to social networks is an issue that is addressed in the reviewed literature. As there are millions of users who are scattered all around the world, finding an optimal way to place and replicate the data related to them in a cost effective way while guaranteeing service level agreements is still a challenge. Social networks data replication in cloud is not addressed using evolutionary algorithms such as a genetic algorithm.

VII. Conclusions and Future Work

Novel use of a genetic algorithm for optimising social media data placement and replication in cloud datacentres is presented in this paper. Comparing to different placement strategies, our proposed algorithm can find the most affordable placement strategy while guaranteeing latency requirement for 99.99% of online social network users. Simulation results on the SNAP Facebook dataset show the effectiveness of the proposed algorithm.

We used the SNAP Facebook dataset for experiments and based on the acceptable size of users. It is assumed that they are located in the US and real locations of Facebook datacentres are considered. We are conducting further experiments for larger datasets with real cloud datacentres such as Amazon. Moreover, social networks have a dynamic and growing nature due to the users' mobility and dynamic activities. In the current work a static data placement and replication in social networks is address which will be extended to be applicable in dynamic environments in the future.

Acknowledgment

This research is partly supported by the Australian Research Council Linkage Projects scheme LP130100324.

References

[1] J. Mcauley and J. Leskovec, "Learning to Discover Social Circles in Ego Networks," *Advances in Neural Information Processing Systems 25 (NIPS)*, 2012, pp. 539-547.

[2] E. Protalinski. (2015). *Facebook Passes 1.55B Monthly Active Users and 1.01B Daily Active Users*. Available: http://venturebeat.com/2015/11/04/facebook-passes-1-55b-monthly-active-users-and-1-01-billion-daily-active-users/

[3] J. Constine. (2012). *How Big is Facebook's Data? 2.5 Billion Pieces of Content and 500+ Terabytes Ingested Every Day*. Available: http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day/

[4] R. Miller. (2012). *Facebook's $1 Billion Data Center Network*. Available: http://www.datacenterknowledge.com/archives/2012/02/02/facebooks-1-billion-data-center-network/

[5] R. Miller. (2013). *How Dropbox Stores Stuff for 200 Million Users*. Available: http://www.datacenterknowledge.com/archives/2013/10/23/how-dropbox-stores-stuff-for-200-million-users/

[6] A. Weiss, "Computing in the clouds," *Computing* 16, 2007, pp. 16-25.

[7] *Advantages and Disadvantages of Cloud Computing*. Available: http://www.levelcloud.net/why-levelcloud/cloud-education-center/advantages-and-disadvantages-of-cloud-computing/

[8] Y. Wu, C. Wu,B. Li, L. Zhang, Z. Li, and F. C. M. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE Conference on Computer Communications (INFOCOM),* 2012, pp. 684-692.

[9] M. Mitchell. (1998). *An introduction to genetic algorithms*. MIT press.

[10] *The Facebook Data Center FAQ*. Available: http://www.datacenterknowledge.com/the-facebook-data-center-faq/

[11] M. Obitko. (1998). *Introduction to Genetic Algorithms*. Available: http://www.obitko.com/tutorials/genetic-algorithms/recommendations.php

[12] X. Liu, J. Chen, and Y. Yang, "A Probabilistic Strategy for Setting Temporal Constraints in Scientific Workflows," *Concurrency and Computation: Practice and Experience, Wiley,* 23(16) , 2011, pp. 1893-1919.

[13] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra*, et al.*, "The little engine (s) that could: scaling online social networks," *ACM SIGCOMM Computer Communication Review* 41(4), 2011, pp. 375-386

[14] D. A. Tran, K. Nguyen, and C. Pham, "S-CLONE: Socially-aware data replication for social networks," *Computer Networks* 56, 2012, pp. 2001-2013.

[15] L. Jiao, J. Li, T. Xu, W. Du, and X. Fu, "Optimizing Cost for Online Social Networks on Geo-Distributed Clouds," *IEEE/ACM Transactions on Networking*, 2014, pp. 99-112.

[16] T. Loukopoulos and I. Ahmad, "Static and Adaptive Distributed Data Replication using Genetic Algorithms," *Journal of Parallel and Distributed Computing* 64(11), 2004, pp. 1270-1285.

[17] T. Phan, K. Ranganathan, and R. Sion, "Evolving Toward the Perfect Schedule: Co-scheduling Job Assignments and Data Replication in Wide-Area Systems Using a Genetic Algorithm," *Job Scheduling Strategies for Parallel Processing* 3834, 2005, pp. 173-193.

[18] W. Guo and X. Wang, "A Data Placement Strategy Based on Genetic Algorithm in Cloud Computing Platform," *10th Web Information System and Application Conference (WISA)*, 2013, pp. 369-372.

[19] Q. Xu, Z. Xu, and T. Wang, "A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing," *International Journal of Intelligence Science* 5, 2015.

[20] Z.I.M. Yusoh and M. Tang, "A penalty-based genetic algorithm for the composite SaaS placement problem in the Cloud," *IEEE Congress on Evolutionary Computation (CEC)* , 2010, pp. 1-8.

[21] G. Liu, H. Shen, and H. Chandler, "Selective Data replication for Online Social Networks with Distributed Datacenters," *21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1-10.

[22] Z. Ye, S. Li, and J. Zhou, "A Two-layer Geo-cloud based Dynamic Replica Creation Strategy," *Applied Mathematics & Information Sciences* 8(1), 2014, pp. 431-440.

[23] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: automated data placement for geo-distributed cloud services," *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[24] M. S. Ardekani and D. B. Terry, "A self-configurable geo-replicated cloud storage system," *11th USENIX conference on Operating Systems Design and Implementation*, 2014.

[25] S. Kadambi, J. Chen, B. F. Cooper, D. Lomax, A. Silberstein, E. Tam*, et al.*, "Where in the World is My Data?," *Very Large Data Base Endowment Inc. (VLDB Endowment) 4(11)*, 2011, pp. 1040-1050.

[26] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "SPANStore: cost-effective geo-replicated storage spanning multiple cloud services," *24th ACM Symposium on Operating Systems Principles*, 2013, pp. 292-308.

[27] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," *IEEE Conference on Computer Communications (INFOCOM),* 2014, pp. 28-36.