Vol.2013-ARC-207 No.34
Vol.2013-HPC-142 No.34
2013/12/17

# Heuristic Data Partitioning for Social Networking Service

Sugianto Angkasa[1,a)] Ryusuke Egawa[2,b)] Hiroyuki Takizawa[1,c)] Hiroaki Kobayashi[2,d)]

***Abstract:*** Managing SNS data is expensive because SNS data have an explosive growth and are highly interconnected. Yet, because of the high interconnectivity of the data, every Read/Write activity of a user is associated with all of his/her friends. The response time for accessing the SNS data generally increases if the data of users and their many connections (friends/followers) are widely located over the network. Most SNS providers are commercial companies and hence need a cost-effective solution to SNS data management. In this paper, we propose a heuristic data partitioning mechanism to store all related data of pairs of users in the same place if they have frequent interaction. Moreover, our mechanism uses activity-based replication. For instance, more replicas are created for active users than inactive users. In performance evaluation against the MySQL random partitioning using real Facebook and Twitter datasets, the proposed heuristic data partitioning and replication mechanism is able to reduce the average response time of the read and write accesses by 53% and by 50%, respectively.

## 1. Introduction

Social Networking Service (SNS) is an online platform that allows users to create public profiles, share highly personalized contents and interact with other users. Unlike traditional online platforms, SNS has a high rate of data accesses. The SNS data size drastically increases with the number of registered users and their user contents. Furthermore, those users' data are strongly related to each other. Considering the data size, it is inevitable to divide these data into some partitions. However, the data are strongly related, and some data might be always accessed along with other data. If those related data are split into different partitions, frequent accesses to multiple partitions via a network increases the response time of the SNS system. Consequently, a data partitioning method is crucial to keep the quality of SNS against the growth of SNS data.

Any activity in SNS is strongly related to user's friends and/or followers that can go beyond tens of millions at the extreme [4]. For instance, a notification is sent to all of his/her friends whenever a Facebook user posts something or SNS user's homepage shows the latest activities of user's and his/her friends. The more connections(friends/followers) SNS users have, the higher the probability of splitting the related data into many different partitions is. Since there are many partitions being accessed, it takes a longer time to complete query execution, which leads to a longer response time. It becomes an even worse problem when random partitioning is implemented.

[1] Graduate School of Information Sciences, Tohoku University
[2] Cyberscience Center, Tohoku University
[a)] sugianto@sc.isc.tohoku.ac.jp
[b)] egawa@isc.tohoku.ac.jp
[c)] tacky@isc.tohoku.ac.jp
[d)] koba@isc.tohoku.ac.jp

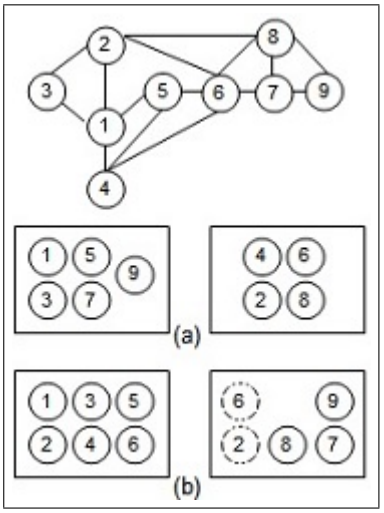


**Fig. 1** (a). random partitioning without replication (b). SPAR with replicas represented as hyphenated circles

A longer response time might be translated into a bad user experience. SNS providers attempt to shorten the response time, however, creating a lot of replicas and/or adding more distributed cache clusters are proven as an expensive solution. Since most SNS providers are started by cash-strapped startup companies, they might not afford the expensive data management solution.

SPAR [1], a Social Partitioning And Replication middleware for SNS, creates an illusion of data locality by putting all related data in one place as shown in **Fig. 1**. The circle and the dashed line circle in figure indicate the master data and replica, respectively. In Fig. 1, data replicas of Users 2 and 6 are created in the right side partition in order to maintain data locality. Because of the data locality, every query is a local query that ensures a shorter query execution time. SPAR claims a faster data accesses performance than the standard MySQL random partitioning. This is be-

cause the standard MySQL random partitioning splits data across many database servers, which are then queried with multiget requests to fetch data of users' friends [1]. Depending on which decision would result in the minimum overall replica creation, the SPAR algorithm decides either to move data or to create a replica. However, SPAR still creates many unused replicas because of two main reasons:

( 1 ) Most of frequently executed queries such as notification are time-dependent [2]. In just a couple of hours or even minutes, the query result might be different;

( 2 ) It assumes that all users must be treated equally. More replicas for active users are effective to improve read accesses performance. However, treating inactive users similarly to active users is wasteful mainly because of the first reason. Inactive users will miss a lot of old notifications.

SPAR might not work well in recent SNS because power users who have a hundred thousand to millions of followers in Twitter or fans in Facebook are no longer rare. In Facebook, users can have no more than 5000 friends, but there is no limitation for fans. It is possible that other SNS would follow Facebook's and Twitter's route to allow unlimited connections because creating a wider network is one of the main purposes of SNS. Hence, storing all related data in the same place becomes nearly impossible. SNS providers need a more practical solution.

To efficiently partition the SNS data, it is important to measure the strength of connection between two users based on their interaction. The friendship or interconnection between pairs of users in the SNS is not equal. SNS users might have more than 100 of friends, but they can only keep in touch with a handful of friends. Even after some periods, the number is still limited because it is rare for SNS users to interact with all of their followers or friends. Therefore, this work proposes a mechanism that stores all related data of two users with frequent interaction in the same place. The related data of pairs of users without frequent interaction may or may not be stored together.

In this work, we also propose a mechanism that classifies users based on their past activities to predict their future activities. There are active users, moderate users and inactive users. A cost-effective data partitioning and replication mechanism for SNS must consider these different types of users. Most of SNS providers' revenues come from online advertisers. The online advertisers are interested more in active users because they have a higher probability of clicking their advertisements than inactive ones. Thus, in data partitioning and replication, a higher priority should be given to the data of active users.

As a whole, this work proposes a heuristic data partitioning and replication mechanism that uses SNS data characteristics to infer future daily activities of each user and user's future interactions for a better SNS data partitioning. Starting from this point, this work will refer to SNS data characteristics as SNS data features. The proposed method also decreases the number of replications without increasing the average Read/Write access response time, while a higher priority is given to data replication of more active users.

The remainder of this paper is organized as follows. Section 2 presents an analysis of several strong features of SNS data using the Facebook and Twitter datasets. Section 3 proposes a heuristic data partitioning and replication mechanism using SNS data features. Section 4 discusses the performance evaluation results of the proposed mechanism. Finally, Section 5 gives conclusions of this work.

## 2. Features of SNS Data

For better data partitioning, it is important to predict the future activities and interactions of each user. In this section, we discuss the following four important features of the SNS data that can be used for the prediction.

- Daily activity
- The number of followers and friends
- Inactive period
- Reply activity

Based on these features, the interactions between two users are measured, and SNS users are classified into three categories; active users, moderate users, and inactive users. By optimizing the data partitioning and replication for active users, we can expect an improvement of the cost efficiency of SNS systems.

### 2.1 Dataset

We collected Twitter dataset that consists of 3,196,611,392 followership links of 109,563,426 users, the information and profiles of those 109,563,426 users, and up to 3,200 most recent tweets from one and half million users. We crawled the followership links instead of the friendship links because the followership graph is much denser. There are two major reasons why we decided to crawl the Twitter data. First, more than 99% of activities on Twitter is to send out a tweet. In addition, most of those tweets are open to the public, and hence we can easily gather the data by using a crawling agent robot. In other SNSs such as Facebook, user activities are more diverse and often are inaccessible to public viewing due to the privacy reasons. Second, the Twitter community is one of the most dense SNS graphs, and keeps growing. For example, top power users, @justinbieber and @ArabicBest, have 46 million of followers and 2 million of friends, respectively.

The other dataset used in this work is the publicly available Facebook New Orleans Network 2009 [3]. The Facebook dataset consists of 90,269 users and 3,646,662 friendship links among those users, 838,092 wall posts timestamp data that spans from September 26th, 2006 to January 22nd, 2009. We used both datasets for the data analysis and peformance evaluation.

### 2.2 Daily Activity

With this feature, SNS users can be classified into three categories, active users, moderate users, and inactive users. In this work, active users are defined as the users who frequently create SNS data. Users who just frequently read the data are not classified to active users because of the following two reasons. One is that SNS providers have to optimize the write access performance rather than the read access performance because a write access is more expensive than a read access, which is shown in [1]. The other reason is that SNS providers can employ distributed cache clusters to improve the read access performance, and hence it is more important to focus on how to improve the write access per-

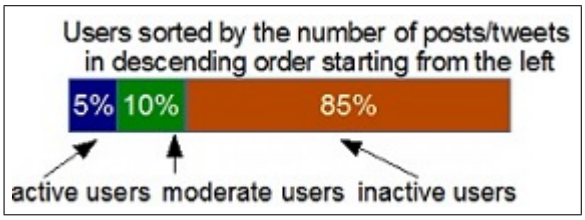


**Fig. 2** Proportion of active users, moderate users and inactive users [12]

**Table 1** Comparison between active, moderate, and inactive user's daily activities

| User | Facebook | | Twitter | |
|---|---|---|---|---|
| Type | % of data | avg. $\bar{a}$/user | % of data | avg. $\bar{a}$/user |
| active | 27.78% | 0.46 posts/day | 19.44% | 82.14 tweets/day |
| moderate | 30.99% | 0.15 posts/day | 38.05% | 13.39 tweets/day |
| inactive | 41.23% | 0.043 posts/day | 42.51% | 1.57 tweets/day |

formance.

According to the observation in [12], the top 5% of Twitter users create almost 75% of tweets. They are active users. On the other hand, 85% of users do not post any tweets at all. Although there is no equivalent observation on Facebook users, this work assumes that the same tendency exists in Facebook. Therefore, in this paper, the populations of active users, moderate users, and inactive users are assumed as in **Fig. 2**.

We define SNS activities as the number of new SNS data, such as tweets in Twitter and the number of wall posts in Facebook. Then, the SNS daily activity is defined by the average daily activity in latest $D$ days:

$$\bar{a} = \frac{\sum_{d=1}^{D} A_d}{D}, \tag{1}$$

where $A_d$ denotes the number of activities that happened in the $d$-th day.

Daily activities of Facebook should include various activities such as wall posting, messaging, chatting, and so on. On the other hand, daily activities of Twitter would consider only tweets and direct messages. The definition of activities can be different for each SNS.

**Table 1** shows the daily activities of three types of users. In our Twitter dataset obtained by crawling the 3,200 latest tweets of 1.5 million users, the active users only contribute to 19.44% of total tweets, not 75% as reported in [12]. This is because the Twitter dataset consists of only 3,200 most recent tweets instead of lifetime tweets. Active users are able to generate 3,200 tweets in just a couple days or weeks. Their lifetime tweets might be tens or hundreds times larger, considering that Twitter has been around for seven years. On the contrary, some of the moderate users and inactive users do not even have 3,200 lifetime tweets. Consequently, it is highly probable that active users might produce more than 19.44% of total data in Twitter.

In Facebook's case, the active users do not generate 75% of total new SNS data too. While the Facebook dataset consists of lifetime wall posts during 2006-2009, the wall post activity is just one of many kinds of activities on Facebook. Active users in Facebook could contribute to more than 27.78% of total data if every activity in Facebook is considered.

From the observation, we found that some of active users on both SNSs are service programs and/or bots. At the extreme, automated bots can generate up to 4,675 tweets per day, and a cumulative of 37,337,696 tweets [4]. Then, we detected at least 49,098 users without any activities at all in our Twitter dataset. There are also 19,302 users without any activities for more than one year. This work assumed that all these 68,400 users are considered dead accounts. After some manual checking to a random subset of dead accounts, it turns out that some users have moved to another account or been banned. There is no need for the SNS system to do anything with these dead accounts until they become active again.

### 2.3 The Number of Followers and Friends

According to our analysis of the SNS data, there is a strong correlation between users' daily activities and the number of followers or friends. An exhaustive study of Twitter shows that 87.4% of Twitter's registered users have less than or equal to 100 followers [13]. According to the same study, users with at least 200 followers covered less than approximately 5% of Twitter's registered users. Following the study in [13], this work assumed that users with more than or equal to 200 followers as users with a lot of followers. On the other hand, users who follow or become friends with at least 300 users are considered as users with a lot of friends. These users represent 5-6% of Twitter's registered users [13]. Users with less than or equal to 100 followers and 100 friends are users with a few of followers and users with a few friends, respectively. Users with a few friends cover as much as 82.2% of Twitter's registered users [13]. In the Facebook case, this work cannot make such generalization because the Facebook dataset covers only one group in Facebook that is New Orleans Network. Hence, to make it similar to the Twitter dataset, we sort the friendship table in descending order and decide that the top 5% and the bottom 87.5% are users with a lot of friends and users with a few of friends, respectively.

Users who have a lot of followers tend to be active users. Around 86.3% of active users are also users with a lot of followers. On the other hand, the percentage of inactive users who also have a few followers is 70.04%.

The number of friends influences users' activities, but not as dominant as the number of followers. Around 81.99% of active users are also users with a lot of friends, but only 26.71% of inactive users are users with a few friends at the same time. In the Facebook dataset, only 42.22% of active users have a lot of friends. However, 99.95% of inactive users have a few of friends. Presumably, users with more followers tend to be active writers, while users with more friends tend to be active readers. Users follow many other users by becoming their friends because they want to read their tweets. Active writers are considered active users in this work.

There are at least 30% of private accounts in the Facebook dataset and up to 9 million of private accounts in the Twitter dataset. These accounts can be accessed only by owners and their connections, and thus the data of these users are unavailable for public crawling. This work also has not finished crawling the latest 3,200 tweets from 109 million users. Because of that, this work needs another way to classify users into one of the three

**Table 2** Medium for accesing SNS [5], [9], [10]

| SNS | desktop | mobile phone |
|---|---|---|
| Facebook | 28.79% | 71.21% |
| Twitter | 25% | 75% |
| LinkedIn | 62% | 38% |

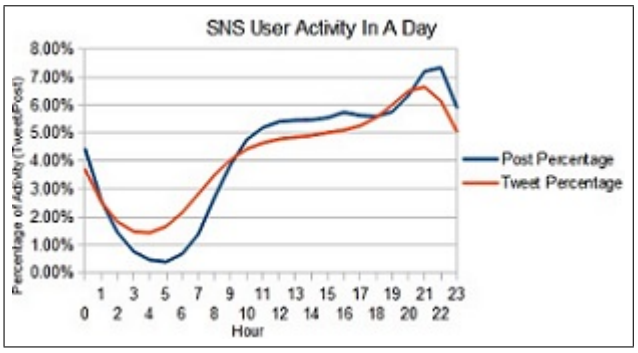


**Fig. 3** Post and tweet percentage in a day

categories. The number of friends or followers feature is used as the alternative way.

### 2.4 Inactive Period

Predicting SNS user activities was easier before the advent of mobile applications. During 2004-2008, it was unlikely for students or workers to be active in SNS during school hours or working hours. SNS users needed to be in front of a computer to do so. Consequently, with some degree of accuracy, predicting their next SNS access time might be possible. However, recent reports [5], [9], [10], [11] have shown that more users access SNS from mobile phones as shown in **Table 2**. More SNS accesses from mobile phones make SNS users' access time become random and unpredictable. Any user might be active in SNS even during school hours or working hours.

Although it has become difficult to predict the active periods of SNS users by the advent of mobile phones, it is easier to predict their inactive periods based on their past inactive periods because they usually need to sleep unless they are automated bots. It has been reported that the percentage of smartphone use becomes the lowest starting a while before the bed time [11].

**Fig. 3** shows that the SNS activities of Facebook and Twitter become the lowest during 1:00 AM to 7:00 AM in each user's time zone. Only 7.68% of posts in Facebook and 13.90% of tweets in Twitter are sent by users during this period. Presumably, this is because the period is a resting time for most of the users. In the Facebook dataset, we converted the unix timestamp of posts from UTC to New Orleans time zone (Central Time Zone). Although there is no user's time zone information in the Facebook dataset, this time zone is used because every user is a member of the New Orleans Network on Facebook. While there exists the possibility of members from different time zones, these users should be a minority. In the Twitter dataset, we converted the tweets' created timestamps from UTC to each user's time zone. Our Twitter dataset spans at least 150 different time zones. Since 33% of users did not specify time zones for their accounts, their time zone information is unavailable. Fig. 3 only shows the tweet activities of 67% Twitter's users that have the time zone information.

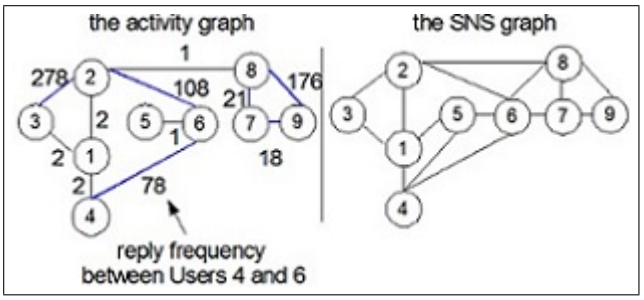


**Fig. 4** Comparison of the activity network with the SNS graph

According to [16], approximately 24% of tweets are created by automated bots. Although the percentage might be different on other SNSs, not a few activities are created by the bots. Most SNS allow to use automated bots mainly for headlines news and news feeding. However, we should preferably improve the response time for human users rather than bots. Hence, SNS providers may need to distinguish bots from human users. As the bots do not have inactive periods, we might be able to distinguish them from human users using the inactive period patterns. SNS users' inactive period information might also helpful for a caching system to eliminate the data candidate to be filled in cache.

### 2.5 Reply Activity

Facebook users on average have 130 friends [15], while Twitter users on average following 102 users [13]. Facebook allows 5000 friends at the maximum, while in Twitter there is no limitation on how many followers a user can have. Regardless of those limitations, SNS users are virtually able to keep in touch with only a small number of friends. The strength of users' connections are different, which is why these connections must be treated differently too. It has been demonstrated in [14] that the strength of connections varies widely, ranging from pairs of users who are best friends to pairs of users who wish they never establish relationship.

In this work, the strength of a link between two users is defined by how frequently they interact. Users connected with a strong link will interact with each other more frequently than users connected with a weak link. Accordingly, the SNS data partitioning and replication need to be considered based on the analysis of SNS users with strong links using a special SNS graph called an activity network. **Fig. 4** shows the difference between the SNS graph and the activity network. In the activity network, each edge has a weight corresponding to the link strength between two users.

Pairs of users with strong links are the first-class citizen during the graph partitioning and must be stored at the same place. A larger weight is given to a stronger link. This can ensure fast query execution while reducing the number of replications. Pairs of users without any communication are obviously connected with weak links. In this work, these edges or links are represented by setting their weights to zero. There are at least two methods to determine the strength between a pair of users. One is the reply frequency between a pair of users. The other is the period length between the first and latest interactions of a pair of users. These two methods are strongly related. Pairs of users spent a couple of

weeks or more to achieve high reply frequencies. Nevertheless, in order to accomodate the new registered users, this work uses the former method.

## 3. Heuristic Data Partitioning and Replication Using SNS Features

This section first defines the link strength between two users by using the activities of unidirectional and bidirectional replies. By using the link strength, we can find the strong links among human users rather than automated bots. Then, this section proposes a heuristic data partitioning and replication mechanism using the SNS data features discussed in Section 2.

### 3.1 Unidirectional and Bidirectional Reply Activity

This work uses the reply frequency between two users to measure the strength of the link between them. If an automated bot generates a lot of reply activities in a short period, it may hide the strong links among human users. Hence, this work focuses on the fact that the reply activities of an automated bot are always unidirectional. The receivers will not reply back because they usually know that the sender are not human users.

This is not the first work that employs reply activities between a pair of users to infer their links strength. However, the previous work [2] does not distinguish the bidirectional and the unidirectional reply activities. Yet, this work realizes the importance to differentiate bidirectional and unidirectional reply activities as the former indicates frequent interaction, while the latter probably not. In this work, bidirectional reply activities might become unidirectional reply activities when the timestamp difference between the closest bidirectional reply activities is too long, e.g., 24 hours. Presumably, those users are starting a new conversation. We still consider unidirectional reply activities because it is better than no activities at all.

To detect the strong links among human users, we define the weight $W_{i,j}$ of every link in latest $D$ days by

$$W_{i,j} = \sum_{d=1}^{D} C \cdot R^{d}_{i,j\leftrightarrow} + R^{d}_{i,j}, \qquad (2)$$

where $C$ is a scaling constant, $R_{i,j\leftrightarrow}$ is the frequency of bidirectional reply activities and $R_{i,j}$ is the frequency of unidirectional reply activities that happened between Users $i$ and $j$ in the $d$-th day. In the next subsection, we will use this metric to detect strong links among human users, and then propose a data partitioning and replication mechanism based on the metric.

### 3.2 Heuristic Data Partitioning

A partitioning method for SNS should store all strongly related data together. It is more feasible to store just strongly related data than all related data in the same place. Pairs of users with high interaction have strongly related data. Their data are frequently accessed together. On the other hand, pairs of users with low or no interactions are rarely accessed together. The data of these users may or may not be stored together.

We propose data partitioning mechanism that uses the reply activity feature to find the weight of each link that is calculated from Eq.(2). Instead of the SNS graph, the proposal partitions the activity network because the network represents the interactions among users. In addition, the activity network has less edges, and hence it requires less computation to find appropriate partitioning. Since there is nothing to do with dead or fake accounts, the activity network becomes even less dense. The proposed heuristic data partitioning produces an activity network that consists of links between a pair of users and its weight. This input is supplied to the METIS library for partitioning. The output of this partitioning is the list of users that must be together in certain partition. For example, Users A and B are listed in the Partition Number 1. The proposal checks whether data of Users A and B are already stored in the Partition Number 1. If it is not, then move them. The implementation of the proposed heuristic mechanism using METIS library will be discussed in Section 4.1.

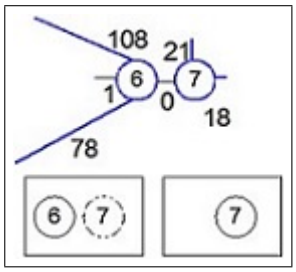


**Fig. 5** Maintaining data locality for a pair of active users with a weak link

This partitioning mechanism is not executed every time the SNS graph changes. This work generates an activity network periodically for periodic SNS data partitioning. Periodic execution still makes sense because while a lot of users join SNS every day, only the friendship/follower graph that changes frequently, but not the activity network graph. A strong link between two users is constructed over the time. While the weight of links might change, most of the time, strong links remain as strong links and weak links remain as weak links. This is because SNS users only keep in touch with a small number of followers and/or friends. In other words, most of SNS users' frequent interactions happen between the same pair of users. There are also cases where an usually close pair of users suddenly stop their interactions and a new pair of users has frequent interaction. It is also assumed that inactive users might not suddenly become active users and conversely active users might not suddenly become inactive users. After some time, SNS users' daily activities will become stable for most of the time. Therefore, a weekly or a monthly data partitioning execution still makes sense.

### 3.3 Heuristic Data Replication

Active users have the most frequently accessed data. Nevertheless, two active users with a weak link might be stored in different partitions because of their low interaction. For example in **Fig. 5**, Users 6 and 7 are active users and connected, but they never interact with each other. Their weak link causes the situation that data of Users 6 and 7 stored in different partition. In such a case, replication to maintain data locality must be created at the partition of user with the bigger data of two (Fig. 5). It is assumed that User 6 has more data, and thus data copy of User 7 are created at the other side. The circle and the dashed line circles in Fig. 5 indicate the master data and replica, respectively. No data locality

is maintained for any other combination because the use of the heuristic partitioning mechanism already allows all related data kept close in the minimum number of partitions. Another reason to give more priority to the active users is to reduce the number of request queues. Completing query execution where the data reside in several partitions takes longer time than local query execution. The possibility of a long waiting queue is high if there is no local query execution for accessing data of active users.

We propose heuristic data replication called activity-based replication here. It maintains data locality for a pair of active users with weak links. The activity-based replication also sets the maximum partitions limit number for all related data of moderate and inactive users. For instance, all of related data to moderate users must not reside in more than four different partitions. In inactive users case, it must not more than eight different partitions. Whenever all related data to moderate or inactive users are split into more than maximum partitions limit number, a replica is created. While this work does not decide the limit number, in general moderate users have low limit number because they also deserve good Quality of Services (QoS). Active users have the best QoS. SNS providers can freely set these limit according to their needs. SNS with very high QoS would use a smaller number of the maximum partitions limit than others.

The average Read/Write access response time for moderate users and inactive users will still be acceptable by the use of distributed cache which is a mandatory for SNS. Furthermore, the data of inactive users are rarely accessed compared to active users. Consequently, the average Read/Write access performance of SNS system is less affected.

While it is better to provide an equal QoS for every user, SNS providers do not have to do so. SNS providers get their revenue from online advertisers and online advertisers value active users more than other kinds of users. Thus, as a cash-limited companies, SNS providers should give more priority for active users by creating more replicas for fast data access.

# 4. Performance Evaluation

The purpose of the following performance evaluation is to demonstrate that the proposal has better Read/Write accesses performance and cost efficiency than the standard MySQL random partitioning. The performance of Read/Write accesses performance is measured by their response time. On the other hand, the cost efficiency is measured by the number of notifications written into database and the average number of created replica.

## 4.1 Implementation

This work uses the METIS library for partitioning the activity network. METIS is a set of serial programs for partitioning graphs, partitioning finite element meshes, and producing fill reducing orderings for sparse matrices. The algorithms implemented in METIS are based on the multilevel recursive-bisection, multilevel k-way, and multi-constraint partitioning schemes. This work arrange the SNS users' data according to the partitioning result. These data are stored into MySQL clusters too, but we disabled its partitioning and replication features. The modified version of the partitioning result from METIS library is also used as a hash table for accessing data in MySQL.

**Table 3** Environment Setup

| node | processor | memory | role |
|---|---|---|---|
| 1 | Intel(R) Atom(TM) @ 1.60GHz | 2GB | management node |
| 2 | Intel(R) Atom(TM) @ 1.60GHz | 2GB | data node |
| 3 | Intel(R) Atom(TM) @ 1.60GHz | 2GB | data node |
| 4 | Dual-Core AMD Opteron @ 2.2Ghz | 6GB | data node |
| 5 | Dual-Core AMD Opteron @ 2.2Ghz | 6GB | data node |
| 6 | Dual-Core AMD Opteron @ 2.2Ghz | 6GB | data node |
| 7 | Dual-Core AMD Opteron @ 2.2Ghz | 6GB | data node |

The SQL data scheme and the SNS system business logic used in this work are taken from Statusnet 1.1.1. It is an open source implementation of SNS. The average daily activities of Facebook or Twitter users can be obtained by eq.(1). In this performance evaluation, we considered only tweets in Twitter and wall posts in Facebook as activities.

## 4.2 Environment Setup

As shown in **Table 3**, the environment for the performance evaluation consists of seven machines connected to the same local area network. We set up a MySQL-5.6.11-ndb-7.3.2-cluster with six partitions on six machines with one machine that is used as a management node. We use the same Facebook and Twitter datasets with the datasets used on SNS data features analysis.

The proposed method is compared to MySQL cluster key partitioning. It is a hash-based random partitioning. Key partitioning is similar to partitioning by hash, except that only one or more columns to be evaluated are supplied, and the MySQL server provides its own hashing function. We supplied primary key and/or unique keys of each table to the MySQL cluster. In MySQL cluster setting, we allow to create only one replica of the SNS data of each user. In our proposal, replicas for moderate users and inactive users are created if their related data stored in more than four and five different partitions, respectively. In a pair of active users with a weak link case, data locality is created by creating replicas. These are the setting for the activity-based replication of proposal in this performance evaluation.

The read and write accesses in this performance evalution emulates how Statusnet deals with user posting activity and user retrieving 20 latest activities from their homepage. In Statusnet, writing a post involves at least three table; table *notice*, *subscription*, and *inbox*. Every time user X sends a post, the post is stored into table *notice* then a notification(*notice_id*) is sent to the inboxes of user X and X's friends. The list user X's friends information is retrieved from table *subscription*. On the other hand, the read access in SNS involves at least two tables that is table *notice* and table *inbox*. SNS homepage shows the latest activities of users and their friends by joining table *notice* and table *inbox*.

We selected 5,000 users randomly from the datasets, in which 20% of users are active, 38% are moderate, 42% are inactive. For each selected user, we sort his/her posts or tweets by their timestamps and then divide them into two. Using the first half of the tweets/posts, initial data partitions are organized by the proposed mechanism and MySQL. Using the second half of the tweets/posts, we simulate new tweets or posts continuously sent to the SNS system. For sending them to the SNS system, we

**Table 4** The response time of MySQL random partitioning and the proposal

| User | Facebook | | Twitter | |
|---|---|---|---|---|
| Type | Read perf. | Write perf. | Read perf. | Write perf. |
| MySQL random partitioning | 4.54ms | 3.3s | 6.01ms | 41.69s |
| The proposal | 1.94ms | 1.42s | 2.83ms | 20.99s |

run six automated bots running on a computer in the same network. These six automated bots emulated six concurrent users using SNS system at the same time. Then, for each selected user, an access to each user's SNS homepage is performed.

### 4.3 Experiment on Write Access Performance

The response time of the write access performance is shown in **Table 4**. In the Facebook dataset case, the proposed method has 57% less response time than MySQL random partitioning. In the Twitter's case, the proposed method reduces MySQL random partitioning response time by 50.3%.

The proposed method shows better write access performance because of two main reasons. The first one is that the proposed method did absolutely nothing to dead accounts. Among the 5,000 randomly selected users' friends , approximate 8.4% of friends in the Facebook case and and 0.073% of friends in the Twitter case are dead accounts. These dead accounts also appear several times as friends of different users from those 5,000 users. Dead or fake accounts exist when users only establish the relationship in SNS with little to no activity at all for a long time. This is a growing problem in SNS and can go as extreme as 20 millions of fake users [9]. In order to minimize this number, SNS providers allow other people to take these accounts. Unlike the proposed method, the MySQL partitioning algorithm is unable to distinguish dead accounts from the others, and thus treats all users alike.

The second reason is that MySQL random partitioning creates one replica copy for all SNS users' data, while the proposed method uses a variable rate number of replicas based on users' daily activities called activity-based replication. The proposal maintains data locality for a pair of active users. On the other hand, the replication is created for moderate users and inactive users only if their related data are split into more than four and five partitions, respectively. On the average, the proposal only creates just portion (57.6%) of replicas created by MySQL random partitioning.

The selected 5,000 users in the Facebook dataset have an average of 42 friends. On the other hand, the selected 5,000 users in the Twitter dataset have an average of 531 followers. In the Twitter case, more notifications are sent because of more connections. This can be considered as the reason the write access performance of Facebook has shorter response time than Twitter.

### 4.4 Experiment on Read Access Performance

As shown in Table 4, the average response time of a read access of the proposal is just 42.73% of the MySQL random partitiong response time in the case of Facebook dataset. In the Twitter dataset, the proposal needs just 47% of response time needed by MySQL random patitioning.

The proposed method shows better read access performance because some of the read accesses are executed locally. Non-local query execution is not too slow because all related data are stored in the minimum number of partitions by the proposed partitioning. The proposed activity-based replication also plays an important role by creating effective replicas.

## 5. Conclusions

In this paper, we have proposed a heuristic data partitioning and replication mechanism that exploits SNS data features in order to predict the user's daily activities and interactions. We have demonstrated that the response time of read and write accesses in our mechanism is at least 53% and 50% less than the MySQL random partitioning. This is because the proposed mechanism exploits the data locality by finding a pair of strongly connected users, and also the use of the activity-based replication. The proposed mechanism is also cost effective because it can reduce the data replica by 42.4% and stores no notification for the dead accounts. Thus, it can save the storage cost without increasing the average response time.

**Acknowledgement** The data analysis and performance evaluation are conducted by using computing resources of Cyberscience Center, Tohoku University.

### References

[1] Pujol, J., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., Rodriguez, P.: The Little Engine(s) That Could: Scaling Online Social Networks, *IEEE/ACM Trans. Netw.*, Vol. 20, No. 4, pp. 1162-1175 (2012).

[2] Yuan, M., Stein, D., Carrasco, B., Trindade, J., Lu, Y.: Partitioning Social Networks for Fast Retrieval of Time-Dependent Queries, *ICDE Workshops.*, pp. 205-212 (2012).

[3] Viswanath, Bimal., Mislove, Alan., Cha, Meeyoung., Gummadi, Krishna P.: On the Evolution of User Interaction in Facebook, *2nd ACM SIGCOMM Workshop Proceedings*, Barcelona, Spain, ACM, WOSN'09 (2009).

[4] Twitter Top 100 Most Followers, Friends, and Tweets (online), available from ⟨http://twittercounter.com/pages/100⟩ (accessed 2013-10-31).

[5] Facebook, Inc.: Facebook Reports Second Quarter 2013 Results (online), available from ⟨http://investor.fb.com/releasedetail.cfm?ReleaseID=780093⟩ (accessed 2013-10-31).

[6] LinkedIn Corporation.: About LinkedIn (online), available from ⟨http://press.linkedin.com/about/ ⟩ (accessed 2013-10-31).

[7] Statistic Brain.: Twitter Statistic (online), available from ⟨http://www.statisticbrain.com/twitter-statistics/ ⟩ (accessed 2013-10-31).

[8] Smith, Craig.: By The Numbers: 87 Amazing Facebook Stats (online), available from ⟨http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/ ⟩ (accessed 2013-10-31).

[9] Smith, Craig.: By The Numbers: 53 Amazing Twitter Stats (online), available from ⟨http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/⟩ (accessed 2013-10-31).

[10] Smith, Craig.: By The Numbers: 41 Amazing LinkedIn Stats (online), available from ⟨http://expandedramblings.com/index.php/by-the-numbers-a-few-important-linkedin-stats/ ⟩ (accessed 2013-10-31).

[11] IDC Research report.: Facebook Always Connected How Smartphones And Social Keep Us Engaged (online), available from ⟨https://fb-public.app.box.com/s/3iq5x6uwnqtq7ki4q8wk⟩ (accessed 2013-10-31).

[12] Sysomos Inc.: Inside Twitter: An In-Depth Look Inside the Twitter World (online), available from ⟨http://www.sysomos.com/insidetwitter/⟩ (accessed 2013-10-31).

[13] Beevolve.: An Exhaustive Study of Twitter Users Across the World (online), available from ⟨http://www.beevolve.com/twitter-statistics/⟩ (accessed

2013-10-31).

[14] Gilbert, Eric and Karahalios, Karrie.: Predicting Tie Strength with Social Media, *CHI '09 Proceedings* , pp. 211–220 (online), DOI: http://doi.acm.org/10.1145/1518701.1518736 (2009).

[15] Statistic Brain.: Facebook Statistic (online), available from ⟨http://www.statisticbrain.com/facebook-statistics/⟩ (accessed 2013-10-31).

[16] Peter Coy.: Could Bots and Spam Smother the Twitter IPO? (online), available from ⟨http://www.businessweek.com/articles/2013-09-25/could-bots-and-spam-smother-the-twitter-ipo⟩ (accessed 2013-10-31).