ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 1, January 2019

Social Partitioning and Replication Middleware for Scaling Online Networks

Beegum Sufiya .J.S[1]

PG Student, Department of Computer Science and Engineering, Sarabhai Institute of Science and Technology, Kerala, India[1]

ABSTRACT: Internet social communities have recently become the dominating challenge of internet application scaling. From social networks such as Facebook and Orkut to mass media services such as YouTube, scalability has already reached a level that no one could have dreamed of a little over a decade ago. Above all others, Google specifically has raised the bar for web application scaling. They host the highest demand and highest storage/computation intensive services all running on their own custom back end management systems known as Big Table, and Google File System (GFS). However, while Google's system has done an amazing job of hosting their current services, they are not necessarily optimal for online social networks. SPAR, on the other hand, is a social partitioning and replication middle-ware designed to optimize social network databases on through a unique approach that is based on the social relationships within the online social network. In this paper I compare the strengths and weaknesses of both Google's solution and SPAR and in addition I propose an alternative solution to take advantage of both.

KEYWORDS: SPAR,Data Management, scaling of online socialnetworks

I. INTRODUCTION

A socialnetworkingservice is an online service, platform, or site that focuses on facilitating the building of social networks or social relations. A social network service consists of a representation of each user (profile), his/her social links, and a variety of additional services. Most social network services are web-based and provide means for users to interact over the Internet, such as e-mail and instant messaging. Online community services are sometimes considered as a social network service, thoughin a broader sense, social network service usually means an individual-centred service whereas online community services are group-centred. Social networking sites allow users to share ideas, activities, events, and interests within their individual networks. The main types of social networking services mean to connect with friends and a recommendation system linked to trust.Online social networks facilitate connections between people based on shared interests, values, membership in particular groups etc. They make it easier for people to find and communicate with individuals who are in their networks using the Web as the interface. There are several different online social networks like Facebook, Twitter and Orkut. Each of these networks has its own unique style, functionality and patterns of usage. Different people are active in these different networks.
The two different types of scaling are:
1.Vertical Scaling
2.Horizontal Scaling

II. RELATED WORK

THERE has been an unprecedented increase in the use of online social networks (OSNs). The most popular OSNs attract hundreds of millions of users (e.g., Facebook), deliver status updates at very high rates (e.g., Twitter), and

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 1, January 2019

distribute user-generated content (UGC) at a global scale (e.g., YouTube). OSNs differ from traditional Web applications on multiple fronts: They handle highly personalized content, encounter non-traditionalworkloads, but

most importantly, deal with highly interconnected data due to the presence of strong community structure among their users All these factors create new challenges for the maintenance, management, and scaling of OSN System. Scaling real systems is hard as it is, but the problem is particularly acute for OSNs due to their astounding growth rate. Twitter, for instance, grew by 1382% between February and March 2009 and was thus forced to redesign and reimplement its architecture several times in order to keep up with the demand. Other OSNs that have failed to do so no longer exist.
An alternative newly proposed system known as SPAR, a social partitioning and replication middle-ware, takes an alternative approach to handling online social networks. Unlike other systems in which the social aspect is managed through the relational schema of the database, SPAR keeps track of the social connections within the network and distributes the data replicas accordingly. Essentially SPAR is a system which ensures that, for example, a user on Facebook has all of their account data stored on one system as well as all of their friend's data on that same system. Their approach is to run as a transparent layer by instead of setting up one giant SQL or Cassandra (Facebook database) as the database and having it be distributed over multiple systems, they set up individual self-contained instances of these databases. The instances are unaware of each other, but each contains all of the data necessary to answer the queries made by a subset of users. SPAR then runs on top of all of these servers and manages what each database contains. When requests come in, SPAR determines which server needs to be used to answer the request and forwards it to that server that contains the request owner's master replica.

III. PROPOSED ALGORITHM

A. *Design Considerations:*

In the proposed system SPAR enables transparent scaling for an OSN by ensuring that all relevant data for a user stays on a machine, thereby maintaining local semantics at the data level and letting all queries be resolved locally on that machine. This creates the illusion of the system running on a single centralized server. It avoids the potential performance problems of having to query multiple servers across a network. The random partitioning solutions used by established OSNs to split data across thousands of database servers, which are then queried with multiget requests to fetch a user's neighbors' data. This can result in unexpected response times, determined by the latency of the worse server, and can be particularly acute under heavy data center loads, where sudden network delays or network congestion can cause performance problems

B. Description of the Proposed Algorithm:

The main modules are Directory Service (DS), the Local Directory Service (LDS), the Partition Manager (PM), and the Replication Manager (RM).

1 Directory Service (DS)

- The data distribution is handled by the Directory Service (DS)
  - Returns the server that hosts the master replica of a user through a key table look-up: $H_m(key) \rightarrow u$
  - DS also resolves the list of all servers where the user has replicas in: $H_s(i) \rightarrow \{u, v, ...,w\}$.
  - Directory Service is implemented as a DHT with consistent caching and is distributed across all servers in the data back end.

2. Local Directory Service (LDS)

- The Local Directory Service (LDS) contains a partial view of the DS.

The LDS only acts as cache of the DS for performance reasons and is invalidated by the DS whenever the location of a replica changes.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 1, January 2019

3. Partition Manager (PM)

- Data partitioning is run by Partition Manager (PM) for the following functions:
    - Map the user's key to its replicas, whether master or slaves,
    - Schedule the movement of replicas, and
    - Re-distribute replicas in the event of server addition or removal.
- SPAR algorithm can allow the *PM* to be distributed.
- *PM* is the only component that can update the *DS*.
- Data Migration
    - After a movement, all replicas undergo reconciliation to avoid inconsistencies that could have arisen during the movement
    - Relies on the semantic reconciliation based on versioning.

4. Replication Manager (RM)

- Data Replication
    - The Replication Manager RM runs on each server of the data back-end.
    - RM propagates the writes that take place on a user's master to all her slaves using an eventual consistency model, which guarantees that all the replicas will – with time – be in sync.

IV. PSEUDO CODE

Step1: The *RM* sits on top of a datastore and controls and modifies the queries.
Step2: The read events (selects) are forwarded directly to the datastore without delay.
Step3: The write events (updates, inserts, deletes) are analysed and can be altered both for replication and performance reasons.
Step4: To illustrate the inner workings, we use an example of an OSN application and a write operation, using MySQL as an example datastore.
Step5: A user wants to create a new event, which generates the following command to be inserted in MySQL: *insert intoevent.*
Step6: where and are the pointers to the user and the event's content. The *RM* will react to this command by obtaining the target table *event*.
Step7: The *RM* knows, through simple configuration rules, that the *event* table is partitioned, and thus the *insert* needs to be replicated to all the servers hosting the slave replicas of user.
Step8: The*RM* queries its *LDS* to obtain the list of servers to propagate the insert and issues the same *insert* command to the local MySQL.
Step9: Additionally, this event will be broadcast to all the neighbours of the user. For each contact of, the application will generate an *insert into event-inbox*.
Step10: finally,*RM* will replicate the same commands to all the appropriate servers.

1) Simulation Results

1.Random Partitioning Algorithm

Key-Value-based stores like Cassandra, HBase, MongoDB, Simple DB, etc., partition data randomly across servers. To the best of our knowledge, random partition is the default used in most commercial systems.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 1, January 2019

7.1.2Graph Partitioning: Used METIS Algorithm

There exist several offline algorithms for partitioning a graph into a fixed number of equal-sized partitions in such a way that the number of interpretations edges gets minimized. We use METIS, which is known to be very fast and yield high-quality partitions for large socialgraphs.

2.Modularity Optimization (MO+) Algorithm

Modified original MO algorithm for fixed number of equal size partition We also consider an offline community detection algorithm built around the modularity metric. We modified it in order to be able to create a fixed number of equal-sized partitions . Our modified version, called MO+, operates by grouping the communities in partitions sequentially until a given partition is full. In the case a community is larger than the predefined size, we apply MO recursively to the community

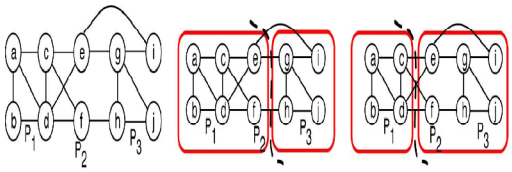

Fig.1. Relationship between two users

V. CONCLUSION

Akey aim of this study was the implementation of a technique SPAR (Social Partitioning and Replication) scaling online social networks with minimum replication. Server addition and Server removal actions can also be implemented by the help of replication manager with high performance. On such a large scale as Google is operating on, performance enhancements such as these could have a drastic overall effect given the incredible high utilization of their services. Even if this merely reduces their network bandwidth usage, it would not only make their response times and throughput faster but would reduce the cost of their currently running systems. Obviously now it would be incredibly hard to make such a major change in a system that has so much usage and where stability is a must. However, given the modern change is internet applications and specifically online social networks, it may be time for all major web hosts to begin using the dense social aspect as an advantage instead of an obstacle.

FUTURE SCOPE

SPAR considers only the social graph. But OSN has different degree of interaction among users. So SPAR like methods should be explored considering the interaction. SPAR did not considered the use of multiple masters sites for each user, though i ammentioned this as a future work.

REFERENCES

[1] High Scalability, “Facebook’s memcachedmultiget hole: More machines != More capacity,” 2009 [Online]. Available: http:// highscalability.com/blog/2009/10/26/facebooks-memcachedmultiget- hole-more-machines-more-capacit.html
[2] High Scalability, “Friendster lost lead due to failure to scale,” 2007 [Online]. Available: http://highscalability.com/blog/2007/11/13/ friendster-lost-lead-because-of-a-failure-to-scale.html
[3] V. Agunati, “Notes From Scaling MySQL—Up or Out,” 2008 [Online]. Available: http://venublog.com/2008/04/16/notes-from-scalingmysql- up-or-out.
[4] RightScale, “RightScale,” 2010 [Online]. Available: http://www. rightscale.com
[5] StatusNet, “Status Net,” 2010 [Online]. Available: http://status.net
[6] N. Niclausse, “Tsung: Distributed load testing tool,” 2010 [Online]. Available: http://tsung.erlang-projects.org.
[7] High Scalability, “Scaling Twitter: Making Twitter 10000 percent faster,” 2009 [Online]. Available: http://highscalability.com/scalingtwitter-making-twitter-10000-percent-faster.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 1, January 2019

[8] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken,\ J. R. Douceur, Jon, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," in *Proc. OSDI*, 2002, pp. 1–14.
[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, 2009.
[10] S. Arora, S. Rao, and U. Vazirani, "Expander flows, geometric embeddings and graph partitioning," *J. ACM*, vol. 56, no. 2, pp. 1–37, 2009.