ISSN:0975-9646

M. Lokesh Kumar Rao et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 7179-7183

Load Balancing For Presence Server Architecture

M. Lokesh Kumar Rao[1]
PG Scholar, Dept. of CSE
MITS Engineering college, JNTUA
Madanapalle, Andhra Pradesh, India

D. Kasi Viswanath[2]
Assistant Professor, Dept. of CSE
MITS Engineering college, JNTUA
Madanapalle, Andhra pradesh, India

***Abstract*- The importance of social network applications on mobile devices is becoming more popular. The mobile presence service is a vital component in social network through which it handles every mobile user information, such as the network address, present status and GPS location, and also updates the user online buddies with the information frequently. When plenty updates occur continuously more number of messages are sent to the servers, this may lead to scalability problem in large scale mobile presence services. To address this, we use a server architecture known as PresenceCloud, which supports large scale social network applications. In day to day life, the number of mobile users of social network applications is increasing rapidly, due to this the response time from the server may be decreased by getting more requests from the mobile users. In our proposed method we are introducing an algorithm for Load balancing, which allocate the work to the clusters of presence server. The load balancing algorithm is Transaction-Least-Work-Left(TLWL), used to allocate work to least values of the servers. This algorithm reduces the response time by distributing the requests to all other servers.**

Keywords- Distributed Presence servers, cloud computing, social networks, load balancing, mobile presence services

I. INTRODUCTION

Social network applications are rapidly growing and increasing numbers of people to communicate and share information on the Internet. At the same time the mobile devices are also becoming more powerful and offering high speed internet services. Due to this, users expecting these social network services to be accessible on their mobile devices. Thus, the importance of social network applications on mobile devices is becoming more popular.

Because of the ubiquity of the internet, the mobile devices and cloud computing environments providing presence enabled applications. Examples of presence enabled applications are WhatsApp [2], Facebook [3], Twitter [4], Viber [5], Foursquare [6], Google Latitude [7], Hike [8] and Mobile Instant Messaging (MIM) [10]. Social network services are changing the ways on the internet in which users engage with their friends. To interact with their friends, the social network services can exploit the information about the status of participants. They enable participants to share live experiences instantly across the world by using wireless mobile network technologies on mobile devices. In the cloud computing environment, a mobile presence service is an important segment of social network services. The key functionality of a mobile presence service is to keep up a breakthrough rundown of data of all portable clients. This data incorporates insights around a portable client's area, action, accessibility, gadget capacity and inclination. In infx`ormal organization benefits every client has a companion rundown or pal rundown , which contains the contact points of interest of all other versatile clients that they need to correspond with. At whatever point client travels from one status the other, the status of the portable client is consequently telecast every individual on the companion list. Most vicinity administrations use server cluster technology [9], to maximize search speed and minimize the time of the administrations.

To backing the tremendous number of clients around the world, numerous web administrations has been conveyed in circulated situations and distributed computing applications. Presencecloud is an adaptable server to server overlay structural engineering which enhances the productivity of portable vicinity administrations and we propose burden adjusting for vicinity servers in Presencecloud to impart the heap among all the vicinity servers. In the first place, we inspect the server construction modeling of existing vicinity benefits and present the mate rundown seek issue in circulated vicinity structural engineering. At t hat point we talk about the outline of Presencecloud, a versatile server building design for portable vicinity administrations. To encourage effective pal rundown looking, Presencecloud sorts out vicinity servers into majority based server to server construction modeling. We investigate execution unpredictability of Presencecloud, cross section based plan and DHT (Distributed Hash Table) based plan. Through recreations, we additionally analyze execution of three methodologies as far as number of messages created, search satisfactoin and buddy notice time.

II. THE PROBLEM STATEMENT

In this area, we portray the framework model and bubby list search problem. Formally, we accept the geologically circulated presence servers to structure a server to server overlay system, G = (V , E), where V is the situated of the presence server hubs and E is a gathering of requested sets of V. Every hub $n_i \in V$ speaks to a vicinity server and a component of E is a couple $(n_i , n_j) \in E$ with $n_i , n_j \in V$. The edge (n_i , n_j) is called a friendly edge of ni and approaching edge of nj . Since the pair is requested, $(n_i , n_j) \in E$ is not equal to $(n_j , n_i) \in E$. We indicate a set of the versatile clients in a vicinity benefit as $U = \{ u_1, \ldots , u_i , \ldots , u_m \}$, where $1 \leq i \leq m$ and m is the quantity of versatile clients.

Definition: Buddy list $B_i = \{ b_1, b_2, \ldots., b_k \}$ of client $u_i \in U$, is characterized as a subset of U, where $0 < k \leq |u|$. Moreover, B is a symmetric connection, i.e., $u_i \in B_i$.

A. Analysis of Naive Architecture of Mobile Presence Service

To search for buddies of recently arrived client in a design, we will gave a dissection of the normal rate of messages produced of mobile presence services. Every mobile client can join and leave the presence service ramdomly, and every mobile client knows those clients specifically appended to it. The likelihood for a mobile client to append to a PS node can be uniform. We should indicate λ the normal arriving rate of mobile clients in a mobile presence service. We expect every PS node to have infinite service capacity. Subsequently, η= ψ/n is the average rate of mobile clients appending to a PS node, where n is meant the number of PS nodes in a mobile presence service. Let h signify the likelihood of having all clients in the mate rundown of ui to be connecting to the same PS node as ui. It is the likelihood of having no compelling reason to send search messages when ui appends to a PS node. In this way,

$$k = \prod_{|Bi|} \frac{1}{n} = n^{-|Bi|}$$

The normal number of search messages produced by this PS node for every unit time is then

$$(n-1) \times (1- k) \times \eta$$

For a sensible size of set Bi (e.g., |bi|≥ 3) and n ≥100, we consider the normal number S of messages produced by the n PS nodes for every unit time, then we have

$$S= n \times (n - 1) \times (1-k) \times \eta$$
$$= n \times (n - 1) \times (1-k) \times \psi/n$$
$$\cong (n - 1) \times \psi$$

Hence, as the number of PS nodes increase, both the correspondence and the communication and the total CPU processing overhead of presence servers also increase. At the point when η increases significantly, is has a real effect on the system overhead. To address this, we propose new load balancing algorithm called Transaction-Least-Work-Left (TLWL), used to allocate work to least values of the servers. This algorithm reduces the response time by distributing the requests to all other servers.

III. PRESECECLOUD SERVER OVERLAY

The construction algorithm of PresenceCloud server overlay organizes the PS nodes into a server to server overlay, which gives us a low diameter property. This property needs only two hops to reach any PS nodes. This construction algorithm of PresenceCloud maintains a PS list of $O(\sqrt{n})$ for each preence server.

A mobile client can get to the web and make an information association with Presencecloud by means of Wifi or 3g services in the mobile internet. For control message transmission the mobile client opens a TCP connection to the presence server. After establishment of control channel the mobile user client sends an appeal to the associated PS hub for buddy list searching.

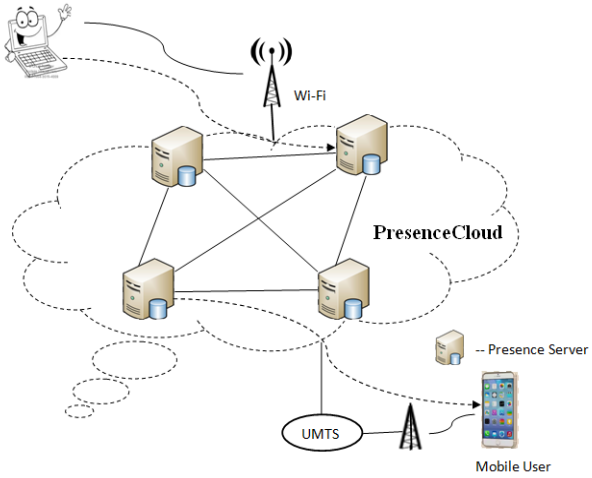

Fig. 1.Architecture for presence cloud

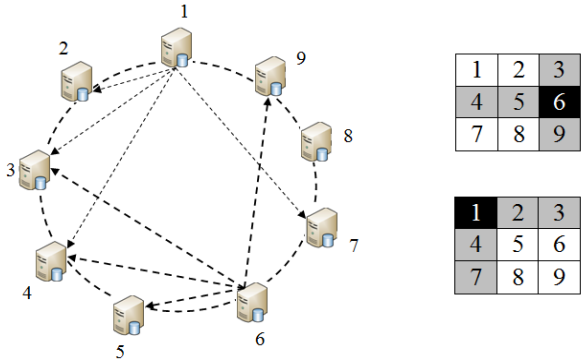

Fig. 2.PresenceCloud Server Overlay

PresenceCloud is focused on the idea of grid quorum system [13], where a presence server node only maintain a set of presence server nodes of size $O(\sqrt{n})$, where where n is the quantity of PS hubs In this framework, every PS hub has a set of PS hubs, called PS list. The size of grid quorum is $[\sqrt{n}] \times [\sqrt{n}]$. The above fig. illustrate a sample of Presencecloud, in which the lattice majority is situated to [√9] × [√9]. In the fig. 2, the PS hub 6 has a PS list {3,4,5,9} and the PS hub 1 has a PS list {2,3,4,7}. In fig. 2, the PS rundown of hub 7 is the situated {1,4,8,9} and one of PS hub 6 is the situated {3,4,5,9}. PS hub 6 can achieve PS hub 7 by k set {4,9}, i.e., a route 6 → 4 → 7 or 6 → 9 →7.

IV. TRANSACTION-LEAST-WORK-LEFT ALGORITHM

To start the explanation of this load balancing algorithm it should be expressed that concerning the mixture showed in the topology used to join presence servers, every presence server has a positive greatest number of neighbors. For example in a framework in which the presence servers are governed by cross mesh topology, every presence server has, at most, four neighbors. We can characterize a field for every presence server the measures of which would focus every one of its neighbor presence servers.

For the purpose of extra clarification, let us assume we call the trademark "direction". As an illustration, in a framework spoke to by fig. 3, for every presence server there are at most four neighbors; in this way "direction" in every presence server can procure four sums i.e. {up, down, left, right} or {1, 2, 3, 4}.

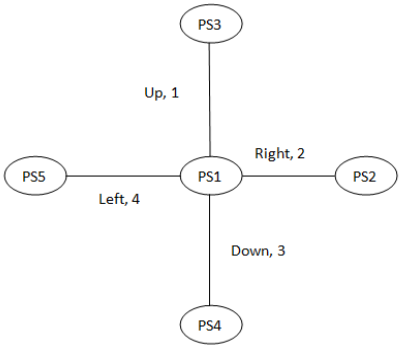

Fig. 3. Particular measures of direction centered on the ps1 presence server

The idea of this process originated from the viewpoint that a presence server might not be idle or over-burden yet have both idle and over-burden presence servers neighboring it and can accordingly serve to relate them; as such the relater presence server can send the message that it is not idle itself however has an idle neighbor presence server to its neighboring presence servers.

The algorithm is as takes after; when a presence server falls idle, it makes an impression on its neighbors. This message incorporates the number of the idle presence server, the message number, a counter and a field to focus the legitimacy of the message. The number of the presence server is, indeed, its id. Any presence server may fall idle many times; to focus the legitimacy of a message i.e. it is not a long ago lapsed message, a message number is utilized. "Counter" is a trademark to which one unit is included each one time a message is passed on starting with one presence server then onto the next and decides the separation of the message from the first idle presence server from which it began.

The nearest recipient presence server spares the message whole with all its connected data and as to the route from which it came. On the off chance that a presence server achieves underloaded level it browses the got messages of nearest presence servers that which has the most priority level (i.e. the closest) and sends the message to its own particular nearest presence servers.

The new load balancing algorithm

If presence server p is idle

if last_msg_num(p, 2)=0 then
last_msg_num(p, 1):=last_msg_num(p, 1)+1;
last_msg_num(p, 2):=1;
for i:=1 to d do
send message to direction i of presence server p(p, last_message_number(p, 1), 1, 1);
end for
end if

If presence server p is underloaded

min:=∞;
walk:=∞;
for i:=1 to d do
if receive_msg(p, i, 4)=1 and receive_msg(p, i, 3) < walk and last_msg_num(receive_msg(p, i, 1),1)=receive_msg(p, i, 2) and last_msg_num(receive_msg(p, i, 1), 2)=1 then
walk:=receive_msg(p, i, 3);
min:=i;
end if
end for
if min≠∞ then
for i:=1 to d do
if i≠min then
send message to direction i of presence server p (receive_message(p, min, 1), receive_msg(p, min, 2), receive_msg(p, min, 3)+1, 1);
end if
end for
end if

If presence server p is overloaded

if match(p)=0 then
min:=∞;
walk:=∞;
for i:=1 to d do
if receive_msg(p, i, 4) = 1 and receive_msg(p, i,3)<walk and last_msg_num(receive_msg(p,i, 1), 1) = receive_msg(p, i, 2) and last_message_number(receive_msg(p, i, 1), 2) =1 then
walk:=receive_msg(p, i, 3);
min:=i;
end if
end for
if min≠∞ then
p1:=p;
p2:=get _presence serverid(p, min);
idle_presence server:=receive_msg(p, min, 1);
d1:=min;
for i:=1 to walk do
receive_msg(p1, d1, 4):=0;
path(p, idle_presence server, i):=d1;
if i≠walk then
p1:=p2;
p2:=−1;
for j:=1 to d do
if receive_msg(p1, j, 4)=1 and receive_message(p1, j,1) = idle_presence server and
last_message_number(idle_presence server, 1) = receive_msg(p1, j, 2) and last_msg_num(idle_presence server, 2) = 1 then
d1:=j;
p2:=get _presenceserverid(p1, d1);
end if
end for
end if
if p2=−1 then
break;
end if

end for
if p2≠−1 and last_msg_num(idle_presence server,2)=1 then
last_msg_num(idle_presence server, 2):=2;
match(p):=idle_presence server;
end if
end if
end if

Inevitably an over-burden presence server browses among its gotten messages the unified with the most priority in order to send an allotment of its heap to the presence server whence the message began.

In this pseudo-code, d demonstrates the maximum number of neighbors that any presence server with in a framework typifying a particular convention can have; n is the quantity of presence servers in the framework. The last_msg_num exhibit of a n*2 cluster in which last_msg_num(p, 1) demonstrates the quantity of the last message sent by presence server p when out of gear status and last_msg_num(p, 2) demonstrates the current status of the message; on the off chance that it be zero the message is no more substantial, if 1 the message is legitimate and if 2 it shows that the message has been gotten by an over-burden presence server that it prepared to exchange burden to the idle presence server. After burden exchange the amount of this field will equivalent zero.

V. Performance Evalution

The related architectures related to our implementation are PresenceCloud, Chord and a Mesh based presence server architecture. We can perform tests up to 20000 users and 2048 Ps nodes by using packet level simulator. To simulate internet networks we apply King topology [16] and Brite topology [17].

The three metrics that are used to measure the performance of server architecture are 1) Total searching messages: This represent the aggregate number of messages exchanged between question initiator and alternate PS hubs. 2) Average searching messages per arrived user: the quantity of looking messages utilized for every arrived client. 3) Average searching latency: This speaks to that normal pal looking time for a portable client.

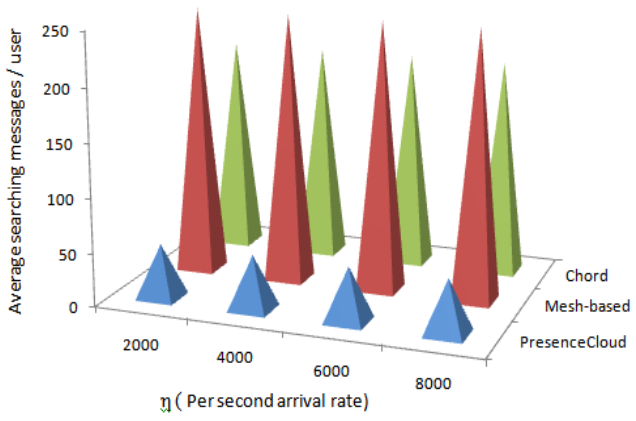

Fig. 4.The normal message transmissions for every seeking operation

The aggregate number of seeking messages is commanded by the client user arrival rate (η) essentially. Chord and mesh-base require substantial number of messages for seeking buddy lists where as PresenceCloud requires few messages. The normal number of seeking message transmissions is free of client entry design. Expanding the rate of client entry design does not expand the normal number of seeking message transmissions.

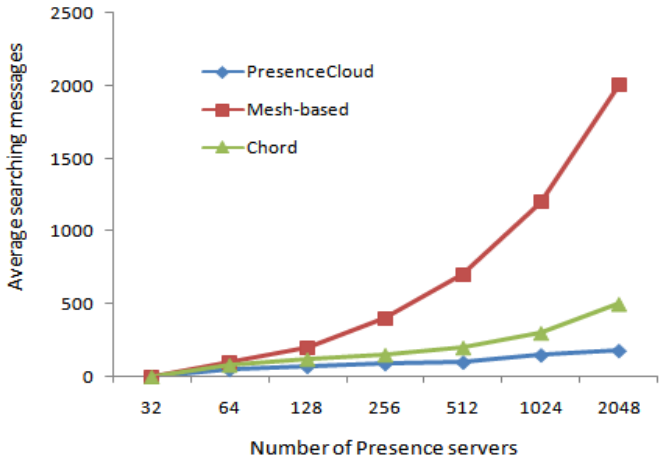

Fig. 5.Normal seeking messages versus number of PS hubs

The above Fig. plots the normal number of seeking messages for every looking operation in different number of PS hubs. The normal message transmissions of Presencecloud increments bit by bit with the quantity of servers. Then again, the normal message transmissions of PresenceCloud is bounded by $4 \times \sqrt{n}$. Mesh-based performs poor than other two designs, it requires $O(\sqrt{n})$ searching complexity. The quantity of normal message transmissions develops gradually with the system measure in Presencecloud and Chord based designs.

VI. Conclusion

In this, we have presented a novel method to load balancing for presence servers. The proposed algorithm reduces the response time by distributing the messages to all other servers and improves the throughput of the system.

References

[1] Chi-Jen Wu, Jan-Ming Ho, "A Scalable Server Architecture for Mobile Presence Services in Social Network Applications", IEEE TRANSACTIONS ON MOBILE COMPUTING,Vol. 12, No. 2, Feb 2013.
[2] Whatsapp, http://www.whatsapp.com
[3] Facebook, http://www.facebook.com
[4] Twitter, http://www.twitter.com
[5] Viber, http://www.viber.com
[6] Foursquare, http://www.foursquare.com
[7] Google Latitude, http://www.google.com/intl/enus/latitude/intro.html
[8] Hike, http://www.hike.com
[9] R.B. Jennings, E.M. Nahum, D.P. Olshefski, D. Saha, Z.-Y. Shae, and C. Waters, "A Study of Internet Instant Messaging and

ChatProtocols," IEEE Network, vol. 20, no. 6, pp. 16-21, July/Aug.2006.
[10] Z. Xiao, L. Guo, and J. Tracey, "Understanding Instant Messaging Traffic Characteristics" Proc. IEEE 27th Int'l Conf. Distributed Computing Systems (ICDCS), 2007.
[11] K. Singh and H. Schulzrinne, "Peer-to-Peer Internet Telephony Using SIP," Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDVA), 2005.
[12] A. Houri, E. Aoki, S. Parameswar, T. Rang, V. Singh, and H. Schulzrinne, "Presence Interdomain Scaling Analysis for SIP/ SIMPLE," IETF Internet draft, 2009.
[13] M. Maekawa, "A $\sqrt{n}$ Algorithm for Mutual Exclusion in Decentralized Systems," ACM Trans. Computer Systems, vol. 3,pp. 145-159, 1985.
[14] Berger, E., Browne, J., 1999. Scalable Load Distribution and Load Balancing for Dynamic Parallel Programs. Proc. Int. Workshop on Cluster-Based Computing, p.1-5.
[15] Lüling, R., Monien, B., Ramme, F., 1991. A Study on Dynamic Load Balancing Algorithms. Proc. 3rd IEEE SPDP, p.686-689.
[16] K.P. Gummadi, S. Saroiu, and S.D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts,"Proc. Second ACM SIGCOMM Workshop Internet measurment (IMW),2002
[17] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation,"Proc. ACM Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS),2001.
[18] R. Cox, A. Muthitacharoen, and R.T. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service,"Proc. First Int'l Workshop Peer-toPeer Systems (IPTPS),2002.
[19] V. Ramasubramanian and E.G. Sirer, "Beehive: 0(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays,"Proc. USENIX First Conf. Symp. Networked Systems Design and Implementation (NSDI),2004.
[20] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model," Proc. Workshop New Security Paradigms,1997.
[21] P. Anick, "Using Terminological Feedback for Web Search Refinement: A Log-Based Study,"Proc. ACM SIGIR Conf. Research and Development in Information Retrieval,pp. 88-95, 2003.
[22] M. Steiner, T. En-Najjary, and E.W. Biersack, "Long Term Study of Peer Behavior in the KAD DHT,"IEEE/ACM Trans. Networking, vol. 17, no. 5, pp. 1371-1384, Oct. 2009.
[23] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H.Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet," IEEE/ACM Tran. Networking, vol. 11, no. 1, pp. 17-32, Feb. 2003.
[24] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence Describes Instant Messaging (IM), the Most Common Application of XMPP," IETF RFC 3921, 2004.
[25] Osman, A., Ammar, H., 2002.Dynamic load balancing strategies for parallel computers. Sci. Ann. J. Cuza Univ., 11:110-120.
[26] Garcia, T., Semé, D., 2006. A Load Balancing Technique for Some Coarse-Grained Multicomputer Algorithms. 21st Int. Conf. on Computers and Their Applications, p.301- 306.
[27] Grama, A., Gupta, A., Karypis, G., Kumar, V., 2003. Introduction to Parallel Computing (2nd Ed.). Addison Wesley, USA.
[28] Peer-to-Peer Session Initiation Protocol IETF Working Group, http://www.ietf.org/html.charters/p2psip-charter.html, 2012.
[29] P. Saint-Andre, "Interdomain Presence Scaling Analysis for the Extensible Messaging and Presence Protocol (XMPP)," IETF Internet draft, 2008.
[30] P. Bellavista, A. Corradi, and L. Foschini, "IMS-Based Presence Service with Enhanced Scalability and Guaranteed QoS for Interdomain Enterprise Mobility," IEEE Wireless Comm., vol. 16, no. 3, pp. 16-23, June 2009.
[31] X. Chen, S. Ren, H. Wang, and X. Zhang, "SCOPE: Scalable Consistency Maintenance in Structured P2P Systems,"Proc. IEEE INFOCOM,2005.
[32] SIP for Instant Messaging and Presence Leveraging Extensions IETF Working Group, http://www.ietf.org/html.charters/ simple-charter.html, 2012.