ISSN 2395-1621

Network Load Balancing

[#1]Miss. Amruta S. Shinde, [#2]Miss. Snehal R. Kumbhar, [#3]Miss. Pallavi M. Patil, [#4]Miss. Surabhi S. Khotkar, [#5]Prof. D.V. Jadhav

[1]ameeshinde@gmail.com
[2]kumbharsnehal273@gmail.com
[3]pallavidcoer@gmail.com
[4]surabhi.khotkar17@gmail.com
[5]deepak.jadhav@zealeducation.com

[#12345]Computer Engineering Department, Zeal College of Engineering and Research, Pune, Maharashtra, India

ABSTRACT

When large number of people at the same time start hitting the web application, it has been observed that web applications hang for example filling university examination form or Scholarship form. The server going down because the load on the server gets increased as they are using the single server. In this concept we are using n numbers of servers. We are also using Discrete diffusive load balancing in which only integer tasks are transferred. The existing system do this by checking load of its own as well as load of adjacent nodes. To reduce the load of main server and balance the network in this project, the concepts of queue and Round Robin Algorithm are applied. The objective of this project is to manage the load on the network server and distribute it in n servers. In this project we have introduced new mechanism for redirecting incoming client requests to the most appropriate server, hence overall system requests loads get balanced. This mechanism leverages local balancing in order to achieve global balancing. This is carried out through a periodic interaction among the system nodes. The node has to check its own load as well as adjacent nodes and due to that reason the existing system is more time consuming. So the aforementioned concept is introduced to increase the efficiency of the system and avoid the servers from crashing.

ARTICLE INFO

Article History

Received :24th May 2016

Received in revised form :

26th May 2016

Accepted : 28th May 2016

Published online :

30th May 2016

Keywords—Discrete diffusive load balancing, Round Robin Algorithm, reallocation.

I. INTRODUCTION

A. Project Idea

When large number of people at the same time start hitting the website, it has been observed that web applications hang for example filling university examination form or Scholarship form. The server going down because the load on the server gets increased as they are using the single server. In this concept we are using n numbers of servers. We are also using Discrete diffusive load balancing in which only integer tasks are transferred. The existing system do this by checking load of its own as well as load of adjacent nodes. Assume processor i, speed of processor si, task xi, the load li is equal to xi/si where si is greater than or equal to 1. if li be the load of node i. If i and j are neighbors in the network and li is greater than lj then node i tries to send to node j a certain number of tasks to be specified by the protocol. Here, we have considered the discrete case where the tasks cannot be divided into smaller sub-tasks. Hence i will have to do some rounding in order to decide how many tasks to send to j.

To reduce the load of main server and balance the network in this project, the concepts of queue and Round Robin Algorithm are applied. The objective of this project is to manage the load on the network server and distribute it in n servers. In this project we have introduced new mechanism for redirecting incoming client requests to the most appropriate server, hence overall system requests loads get balanced. This mechanism leverages local balancing in order to achieve global balancing. This is carried out through a periodic interaction among the system nodes.

B. Goals and Objective

The objective of this project is to balance the load equally in the network and reduce load discrepancy. To design a system to balance the load in the network, in order to increase the stability of the network.

C. Motivation

We have seen that the scholarship website gets hanged when number of students starts hitting the website. As they are using the single server, the load on the server gets increased resulting the server down. So, we are proposing a concept in which we will use 3 or n number of servers. Using concepts of Queue and Round Robin Algorithm we are going to reduce the load of main server.

II. LITERATURE SURVEY

A. Clemens P. J. Adolph, Petra Berenbrink, "Improved Bounds for Discrete Diffusive Load Balancing

In this paper we studied, diffusion load balancing works in rounds. The processors compare their own load with the load of their neighbors. And using their local information only to balance the load with the neighbors, in every round.

B. Thomas Sauerwald and He Sun, "Tight Bounds for Randomized Load Balancing on Arbitrary Network Topologies"

The studies on load balancing assume that the load is arbitrarily divisible. In this so-called continuous case, the diffusion scheme corresponds to a Markov chain on the graph and one can resort to a battery of established techniques to analyze the convergence speed. In particular, the spectral gap captures the time to reach a small discrepancy quite accurately. This relation continues to hold for the matching model, even if the matching's are generated randomly, which might be necessary for graphs with no canonical matching.

C. P. Berenbrink, M. Hoefer, and T. Sauerwald, "Distributed selfish load balancing on networks"

The vertices represent n identical machines in this model in a network and m greater than n selfish agents that unilaterally decide to move from one machine to another. Several protocols are presented for concurrent migration that satisfy desirable properties such as being based only on local information and computation and the absence of global coordination or cooperation of agents. To show rapid convergence of the resulting migration process to states that satisfy different stability or balance criteria is the main contribution.

D. P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrichand T. Sauerwald, "

Randomized diffusion for indivisible loads" For balancing indivisible tasks (tokens) on a network a new randomized diffusion-based algorithm is presented. Minimize the discrepancy between the maximum and minimum load is the aim. The working of algorithm as follows. Every vertex distributes its tokens among its neighbors and itself as evenly as possible. The vertex redistributes its excess tokens among all its neighbors randomly without replacement, if this is not possible without splitting some tokens.

III. SYSTEM ARCHITECTURE

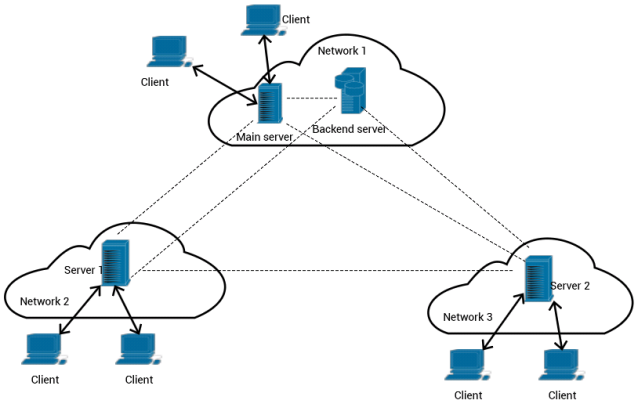


Fig1. System Architecture

Here, we are going to use three servers instead of a single server. The application will be installed on all the three servers. There will be one main server amongst all servers. When the client sends requests to the server, the request will be passed to the main server through the application. The application checks the load of the main server. Each server is assigned with one threshold value. If the load of the main server is less than the threshold value, the request will be passed to the main server. If the load of the main server is more than the threshold value then the request will be passed to next server i.e. server1. Server1 again checks its own load and if it is greater than the threshold value then request will be passed to the next server i.e. server2. Server2 again checks its own load, if it is more then it will pass the request to the server which is in queue. The queue will contain the servers arranged using round robin algorithm.

IV. RELEVANT MATHEMATICS

The computer network is represented as an undirected graph $G = (V, E)$, with vertices represents the nodes and edges represents the links between the adjacent nodes.

If "*a*" and "*b*" are two adjacent nodes and there is an edge between those two nodes.

Assume $[load_a(x) - load_b(x)] > (1/speed_b)$.

Where,

$load_a(x)$ = *load on node "a" at state "x".*

$load_b(x)$ = *load on node "b" at state "x".*

$speed_b$ = *speed of node "b".*

$speed_a$ = *speed of node "a".*

degree (a)=degree of "a".

degree (b)=degree of "b".

$degree_{ab}$ = max{ deg (a), deg (b)}.

Then, "a" transfers $f_{ab}(x)$ number of tasks to node "b".

Where,

$f_{ab}(x)=[load_a(x)-load_b(x)]/ \{\alpha.degree_{ab}.[(1/speed_a)+(1/speed_b)]\}$.

α is set to $4s_{max}$.

Where,

$s_{max} = max_{i \epsilon V}\ speed_i$.

Let "*IP*" be the *input set* and "*OP*" be the *output set*.

IP = { $load_i$ (x), $task_i$(x), $speed_i$(x), $load_j$(x), $task_j$(x), $speed_j$(x), $load_k$(x), $task_k$(x), $speed_k$(x)}

Where,

$load_i$ (x) = load of node "i" at state "x",
$task_i$(x) = number of tasks on node "i" at state "x",
$speed_i$ = speed of node "i",
$load_j$(x) = load of node "j" at state "x",
$task_j$(x) = number of tasks on node "j" at state "x",
$speed_j$(x) = speed of node "j",
$load_k$(x) = load of node "k" at state "x"
$task_k$(x) = number of tasks on node "k" at state "x"
$speed_k$(x) = speed of node "k"

if($[load_i(x) – load_j(x)] > (1/s_j)$)
then node "i" transfers $f_{ij}(x)$ number of tasks to node "j".

if($[load_i(x) – load_k(x)] > (1/s_k)$)
then node "j" transfers $f_{jk}(x)$ number of tasks to node "k".

if($[load_k(x) – load_i(x)] > (1/s_i)$)
then node "k" transfers $f_{ki}(x)$ number of tasks to node "i".
OP = {f_{ij}(x), f_{jk}(x), f_{ki}(x)}

Thus, the above processing can be summarized as follows:

A. **Input:**
Number of tasks on current node and its adjacent node.
Load on the current and its adjacent node.
Speed of the current and its adjacent node

B. **Processing:**
Load on the nodes is calculated and excess load is transferred to the appropriate node.

C. **Output:**
Number of tasks transferred to appropriate node.

V. RESULTS

A. Initial Server CPU Details

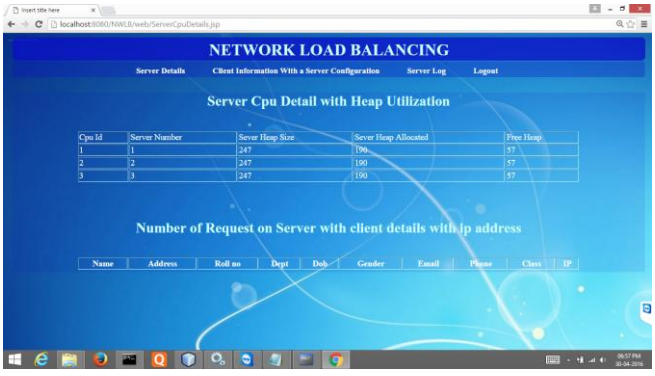


Fig 2. Initial Server CPU Details

B. Server 1 Details

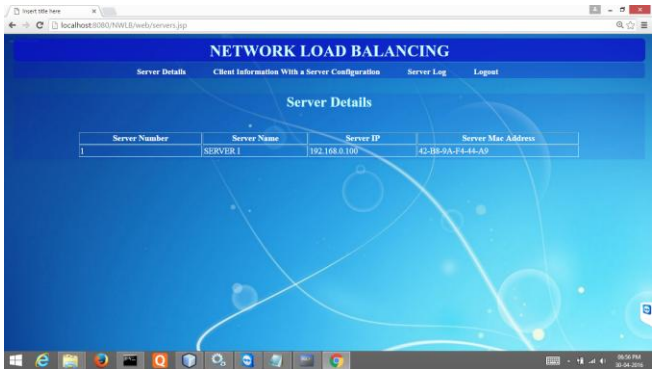


Fig 3. Server 1 Details

C. Server 1 Details After 1st Entry

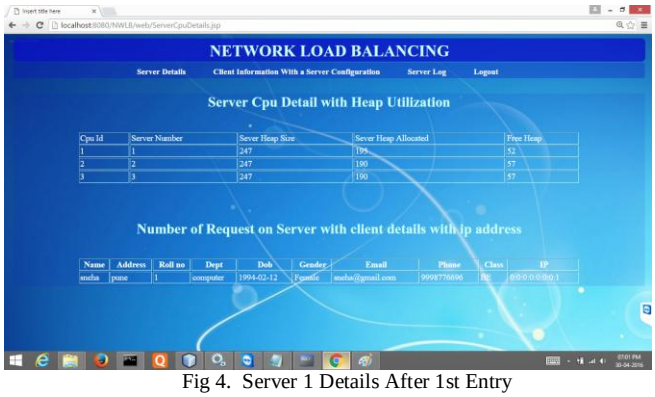
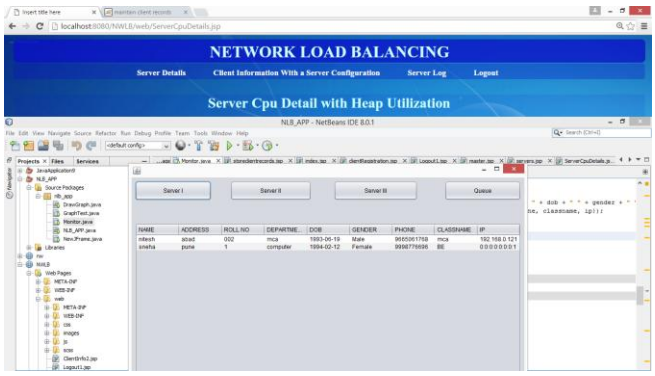


Fig 4. Server 1 Details After 1st Entry

D. Server 1 Log (Entries in Server 1)



E. Fig 5. Server 1 Log (Entries in Server 1)

VI. TESTING

A. To check the functionality of Login Button

Test cases	Required result	Actual result	Status (Pass/Fail)
Check for valid username and password	It should allow the user to proceed.	allow the user to proceed	Pass
Check for invalid username and	By entering the invalid username it should print	print the message please enter the valid	Pass

password	the message please enter the valid username.	username and password.	

B. Check Registration Page for invalid and valid input

Test cases	Required result	Actual result	Status (Pass/Fail)
Check for valid username, password, E-mail Id, Contact.	It should allow the user to proceed.	allow the user to proceed	Pass
Check for valid username, password, E-mail I and invalid , Contact.	By entering the invalid contact it should print the message please enter the valid contact and It should be 10 digits number.	print the message please enter the valid contact and It should be 10 digits number.	Pass
Check for valid username, password, contact and invalid , E-mail id.	By entering the invalid e-mail id it should print the message please enter the valid e-mail id.	print the message please enter the valid e-mail id.	Pass

C. To check the functionality of Register Button

Test cases	Required result	Actual result	Status (Pass/Fail)
Keep any field remain blank/unfilled	Error message "Field;s cannot be empty." Should be displayed	Shows error message "Fields cannot be empty."	Pass
Pass Valid input	Successful Registration	Register Successfully	Pass

D. To check if request is forwarded to next server.

Test cases	Required result	Actual result	Status (Pass/Fail)
Check if server is activated	Request is forwarded to next server	Request is forwarded to Server2.	Pass

VII. CONCLUSION

Network load balancers play an important role in the server rooms. Millions of users hit web applications at the same time. This leads to the web application going down or crashing of the server. This problem arises due to imbalance among the servers. To avoid this problem, network load balancers are used in the server rooms. Load balancer checks which server has more load i.e. which server is imbalanced and forwards its pending client requests to the server which has lesser load. Discrete diffusive load balancing technique is used here, in which only integer tasks are transferred to the next server. Proposed system uses Round Robin and queue. The objective of this project is to manage the load on the network server and distribute it among all the servers. A new mechanism is presented for redirecting incoming client requests to the most appropriate server thus balancing the overall system requests load. This mechanism leverages load balancing in order to achieve global balancing. Network load balancers help us to manage the network efficiently and stabilizing the network.

ACKNOWLEDGEMENT

I take this opportunity to thank my Project guide **Prof. Jadhav D. V.** and Head of the Department **Prof. Sangve S. M.** for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this report. I also thankful to all the Faculty members of Computer Engineering Department for their valuable time, support, comments, suggestions and persuasion. I would also like to thank the institute for providing the required facilities, Internet access and important books.

REFRENCES

[1] Clemens P. J. Adolph, Petra Berenbrink, "Improved Bounds for Discrete Diffusive Load Balancing," in 26th International Parallel and Distributed Processing Symposium IEEE Computer Society, 2012.

[2] Thomas Sauerwald and He Sun, "Tight Bounds for Randomized Load Balancing on Arbitrary Network Topologies," in 53rd Annual Symposium on Foundations of Computer Science IEEE Computer Soc., 2012.

[4] P. Berenbrink, M. Hoefer, and T. Sauerwald, "Distributed selfish load balancing on networks," in Proceedings of 22nd Symposium on Discrete Algorithms (SODA)(2011,to appear), 2011, pp. 1487–1497. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Distributed+Selfish+Load+Balancing+on+Networks#0

[5] C. P. J. Adolphs and P. Berenbrink, "Distributed selfish load balancing with weights and speeds," submitted to STACS 2012. [Online]. Available: http://arxiv.org/abs/1109.6925

[6] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," Journal of Parallel and DistributedComputing, vol. 7, no. 2, pp. 279–301, Oct. 1989.

[7] J. E. Boillat, "Load balancing and Poisson equation in a graph," Concurrency: Practice and Experience, vol. 2,no. 4,

pp. 289–313, Dec. 1990. [Online]. Available: http://doi.wiley.com/10.1002/cpe.4330020403

[8] R. Subramanian and I. D. Scherson, "An analysis of diffusive load-balancing," in Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures - SPAA '94. New York, New York, USA: ACM Press, Aug. 1994, pp. 220–225. [Online]. Available: http://dl.acm.org/citation.cfm?id=181014.181361

[9] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load balancing in dynamic structured peer-to-peer systems," Performance Evaluation, 63(3):217–240, 2006.

