INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY ADVANCED RESEARCH TRENDS
ISSN : 2349-7408
VOLUME 1, ISSUE 2, SEPTEMBER 2014

OPTIMAL DYNAMIC LOAD BALANCE IN DISTRIBUTED SYSTEMS FOR CLIENT SERVER ASSIGNMENT

D.SARITHA
Department of CS&SE, Andhra University Visakhapatnam, Andhra Pradesh

Ch. SATYANANDA REDDY
Professor, Department of CS&SE, Andhra University Visakhapatnam, A.P.

ABSTRACT

Internet is a network of several distributed systems that consists of clients and servers communicating with each other directly or indirectly. In order to improve the performance of such a system, client-server assignment plays an important role. Achieving optimal client-server assignment in internet distributed systems is a great challenge. It is dependent on various factors and can be achieved by various means. Our approach is mainly dependent on two important factors, 1) Total Message length and 2) Available servers in a system and size of the channels through a server. In our approach we propose an algorithm that is based on dynamic method, to obtain an approximately optimal solution for the client-server assignment problem. Our simulation results indicate that the proposed algorithm outperforms other client server assignment algorithms in terms of optimal value and running time.

Keywords: Distributed systems, client-server systems, load balancing, communication load, optimization, dynamic method.

1. INTRODUCTION

The modern internet is a collection of interconnected networks of various systems that share resources. An ideal distributed system provides every node with equal responsibility, and nodes are similar in terms of resource and computational power. However in real world scenarios it is difficult to achieve it as such systems includes overhead of coordinating nodes, resulting in lower performance. Typical distributed system consists of servers and clients and servers are more computational and resource powerful than clients. Typical examples of such systems are [1] e-mail, instant messaging, e-commerce, etc. Communications between two nodes happen through intermediate servers. When node A sends mail to another node B, communication first flows from A to its email server. Email server is responsible for receiving and sending emails, from and for the clients assigned to it. Node A's email server sends data to node B's email server, which in turn is responsible for sending mail to node B. Email servers

communicate with each other on behalf of their client nodes. Clients are assigned to a server based on various parameters like organizations, domains, etc. In our paper we solve client-server assignment problem based on total message length and number of servers available in a given system

Client server assignment can be designed based on the following observations;

1. If two clients are assigned to the same server, the server will receive message from one client and forward to another one. If they are on different servers, the sender client first sends to its server. The sender's server will receive data and forward it to the receiver's server. The receiver server will receive data and forward it to receiver client. Thus the total communication between two frequently interacting clients increases if they are assigned to two different servers. Assigning these two clients to a single server makes all information exchange locally. It is more efficient to have all the clients assigned to few servers to minimize total communication.

2. On the contrary having fewer servers, results in those servers being heavily loaded while others are left underutilized. If a server is heavily loaded it results in low performance due to excessive resource usage on that server. Thus it is necessary to consider load balance on the server while assigning clients [2].

Taking in consideration the current world scenario, client-server assignment problem can be used for a number of applications from social networking, online auctioning to ecommerce.

The rest of this paper organized as follows: section 2 formulates related work in which advantages and disadvantages are there. Section3 formulates problem formulation, section4 formulates proposed method, section5 formulates implementation and evaluation results, section6 formulates conclusion and section 6 formulates conclusion and section7 formulates references.

2. RELATED WORK

The client-server assignment problem can be viewed as an instance of the clustering problem. Deng et al. [3] introduce an efficient graph clustering algorithm called Graclus which utilizes the equivalence between kernel k-means [4] and other graph clustering algorithms including the NC. The Graclus eliminates the time-consuming calculation of eigenvectors inherent

in the NC. Similarly to Metis [5], its three-step "coarsening base clustering-refining" multilevel process enables more "balanced" clustering, and as a result obtains better (smaller) objective values than the NC.

Lang [6] examines some balanced clustering algorithms for power-law graphs. Interestingly, Lang [6] simulates with a graph based on the buddy lists of Yahoo IM, which is also an instant messenger system, and concludes that the combination of solving a semidefinite program and multiple tries of a randomized flow-based rounding methods yields effective results. However, similar to the Graclus, it focuses on balancing each group size. (Bansal et al., 2002) show that exact minimization is NP-complete, and also provide constant-factor approximation algorithm for the problem of minimizing the number of disagreements. Shi and Malik [3] propose the "normalized cut" criterion which minimizes the weight of a cut subject to normalizing terms to prevent unbalanced cuts. The resulting combinatorial problem is relaxed using methods from spectral graph theory. Ta and Zhou [7] take into account of extra interserver communication caused by different clientserver assignments.

3. PROBLEM FORMULATION

In client server assignment, clients are assigned to servers in two different ways.

1. Assigning all clients to one server is impossible due to overloading and completely loses the load balance and also heavily loaded server exhibits a low performance.

2. Assigning clients to different servers is minimizing the overall communication load.

In existing method Gini Coefficient and entropy methods [2] are minimizing the communication load on servers. But it takes more time to check the load balance on every server at each time. It fails in terms of time complexity and memory usage.

To overcome the problem, I proposed a method is dynamic method in which check the load balance on servers at a time based on the total message length and available servers at a time.

4. PROPOSED METHOD

In order to overcome the client server assignment problems, proposed a dynamic method that is used for optimizing the performance of distributed systems over the internet. Such a system consists of a large number of clients who communicate with each other indirectly via a number of intermediate servers. Optimizing the overall performance of such a system then can be formulated as a client server assignment problem whose aim is to assign all the clients to the servers based on total message length and number of available servers in a given system.

In Optimal client server assignment, calculate the expressions for communication load based on given communication pattern among the clients.

Some notations used in this paper are

M: number of servers in the system

N: number of clients in the system, N>M

||A||: Sum of all elements in A.

S: The communication patterns among the clients. $S_{i,j}$ represents the rate of messages sent from clients i to j in the system.

X: the client-server assignment.

Let $P_{s,t}$ represent the rate of messages sent from servers s to t, then

$$P_{s,t} = \sum_{i=1}^{N}\sum_{j=1}^{N} S_{i,j} X_{i,s} X_{j,t} \qquad (2)$$

i.e. $P = X^T SX$ (3)

Communication load is performed on two different types of message passing, they are

1. Intraserver communication i.e. two clients are assigned to the same server, the amount of communication load on the server is 1*(the size of the messages).

2. Interserver communication i.e. the amount of communication load is 1*(the size of the messages for each server) and 2*(the size of the messages) in total.

The communication load for 1) is proportional to $P_{s,s}$. The communication load for 2) is proportional to $P_{s,t}+ P_{t,s}$ (for each server of s and t) because both sending and receiving causes data processing. Then the load generated by the message exchanges is

$$L = P + P^T - P^D \qquad (4)$$

Where P^T is the transpose of P and P^D is the diagonal Matrix of P. Since

$$P = X^T SX, P^T = (X^T SX)^T = X^T S^T X$$

$P^D = (P + P^T)^D / 2 = (X^T SX + X^T S^T X)^D / 2$ We have

$$L = X^T SX + X^T S^T X - \frac{1}{2}(X^T SX + X^T S^T X)^D \qquad (5)$$

Let $A = S + S^T$, then we get

$$Q = X^T AX \qquad (6)$$

$$L = Q - \frac{1}{2}Q^D \qquad (7)$$

$A(\in [0,1]^{N\times N})$ *is symmetric and* $A_{i,j} = A_{j,i}$ can be represented as rate of messages exchanged.

Let $l \in [0,1]^M$ be a vector denoting the communication load for M servers.

Total communication load is defined as total load on all servers

$$\| L \|_1 = \| Q - \frac{1}{2}Q^D \|_1 = \| \frac{1}{2}Q + \frac{1}{2}(Q - Q^D \|_1 \qquad (8)$$

$$= 1 + \| \frac{1}{2}(Q - Q^D) \|_1 = 1 + \sum_{s=1}^{M}\sum_{t=1}^{s=1} L_{s,t} \qquad (9)$$

Let, $F_c = \sum_{s=1}^{M}\sum_{l=1}^{s-1} L_{s,t}$ (10)

F_c represents the amount of Interserver communication that is extra load caused by distributing the servers. The total communication load for Interserver communication is $1+ F_c$. The total communication load for Interserver communication is 1.

In dynamic method, first, check whether a given system is intraserver communication or Interserver communication based on eq (10). If F_c is a zero then it is a intraserver communication otherwise it is a Interserver communication. If a given system is intraserver communication then overload occurs add the server directly. If it is an Interserver communication, first calculate the length of message and check for capacity of channel through that server. If message length is greater than channel size, then check for availability of servers in a given system. Divide the message length into sub parts depending on number of servers available in a system. If server is available then assign message sub part to server. If no servers are there in a given system add another server and assign the client. In this way assign clients to the server and utilize all the available servers in a system.

Algorithm for dynamic method used in Interserver communication:

1. M=Length of message.
2. C=Capacity of channel through servers.
3. S=Number of available servers.
4. Check M>=C
5. If high
6. Then
7. Check S[n] and C[n]
// Now apply dynamic method
8. Divide M1=M/S
9. For i=1 to n
10. If S[i] =true;
11. Then
12. C[i] =M1;
13. Else
14. Add S
15.Repeat from step 1.

Algorithm for dynamic method used in Interserver communication:

1. M=Length of message.
2. C=Capacity of channel through server
3. S=server
4. Check M>=C
5. If high
6. Then
7. Add another server
8. Divide the message length by 2.
9. Assign each sub message to two servers
10. Exit.

5. IMPLEMENTATION AND EVALUATION

5.1Tools

The implementation of the proposed system is evaluated by using the OPNET simulator; It is a network simulator that provides virtual network communication environment. OPNET modeler provides virtual real time environment with GUI. It is easy for understanding the network behavior in various constraints. It is very flexible and easy graphical interface to view the results.

5.2 Simulation Results and Analysis

Simulation results show that the proposed system performs better than the existing system. To show the performance of proposed algorithm, compare the results using without load balance and with load balance.

In experimental analysis, check the percentage of utilization by CPU using without load balance and with load balance.

CPU UTILIZATION (%)

	Without load balance	With load balance
Server 1	26	7
Server 2	0	6
Server 3	0	8

Interpretation of results: In X-axis time is taken and in Y-axis percentage of utilization in CPU is taken as input. In fig1 shows that without load balance. Only server1 is used, server2 and server3 are not used. In fig2 server2 and server 3 are same, because load balance is not applied. After applying the load balance on servers, get the results like in fig 5, 6, 7.

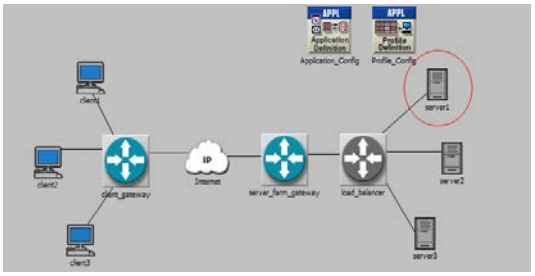

Fig1 Without load balance

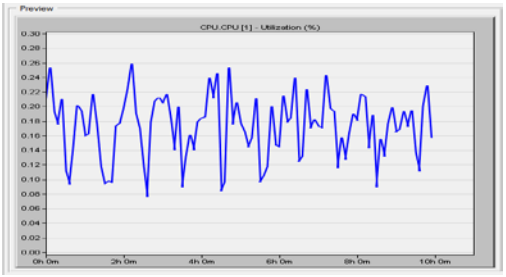

Fig2 Output of server1 without load balance

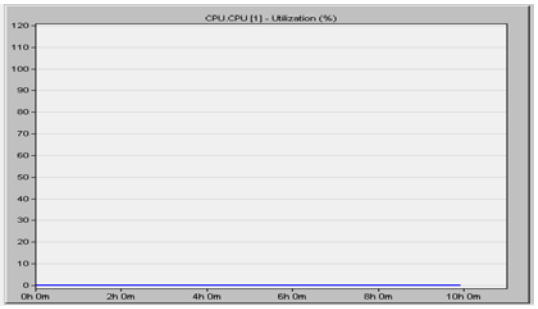

Fig3 Output of server2 and server3

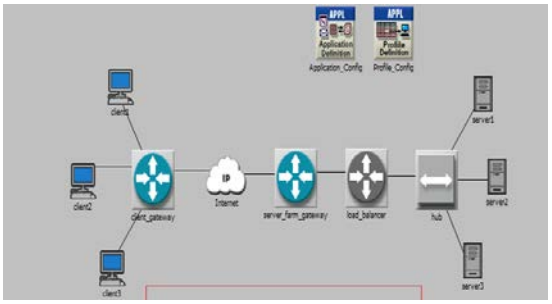

Fig4 With load balance

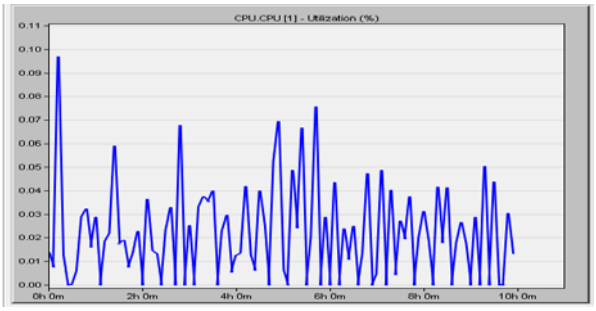

Fig 5 Output of server1 with load balance

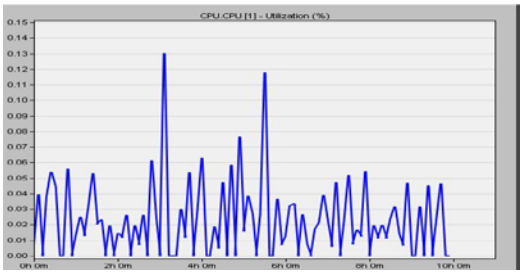

Fig 6 Output of server2 with load balance

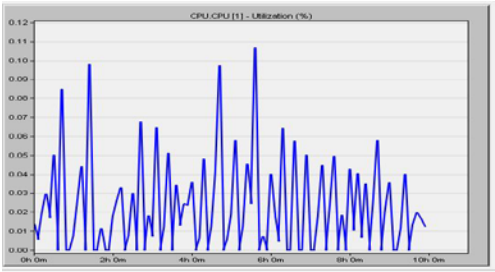

Fig.7 Output of server3 with load balance

6. CONCLUSION

Optimizing the client server assignment is based on Gini Coefficient and entropy has a problem in terms of time complexity and memory usage. It takes more time to check the load balance on every server at each time.

To overcome the problem, I proposed a method is based on some prespecified requirements is on total message length and number of servers available in a system at a time. It presents an algorithmic solution to the client server assignment problem in distributed systems. Our simulation results shows that utilization of CPU by using without load balance on server1 is 26 and utilization of CPU by using with load balance on average of three servers is 7. So the proposed system always finds the optimal solution in terms of optimal value and response time.

7. REFERENCES

1. Optimal Client-Server Assignment for Internet Distributed Systems. Hiroshi Nishida, Member, IEEE, and Thinh Nguyen, Member, IEEE.
2. An Efficient Client Server Assignment for Internet Distributed Systems Swathi Balakrishna, Dr. Ling Ding Computer Science and Systems, University of Washington, Tacoma

3. I. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts Without Eigenvectors a Multilevel Approach," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 11, pp. 1944-1957, Nov. 2007..
4. B. Scho¨lkopf, A. Smola, and K.-R. Mu¨ ller, "Nonlinear Component Analysis as a Kernel Eigen value Problem," Neural Computation, vol. 10, pp. 1299-1319, July 1998
5. G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," SIAM J. Scientific Computing, vol. 20, pp. 359-392, Dec. 1998.
6. K. Lang, "Finding Good Nearly Balanced Cuts in Power Law Graphs," technical report, Yahoo Research Labs, 2004.
7. D.N.B. Ta and S. Zhou, "Efficient Client-To-Server Assignments for Distributed Virtual Environments," Proc. 20th Int'l Conf. Parallel and Distributed Processing (IPDPS '06), 2006.