Improved Method to Solve the Client Server Assignment Problem In Distributed System

Bharati Patil[1], Prof. V.S. Wadne[2]
[1]*Research Scholar, Computer Engineering Department, Pune University*
JSPM's Imperial college of Engg. & Research, Wagholi, Pune,India.
[1]c2patil.s@gmail.com

[2]*Assistant Professor,Computer Engineering Department, Pune University*
JSPM's Imperial College of Engg& Research, Wagholi, Pune, India
[2]*vinods1111@gmail.com*

Abstract:-In distributed system numbers of servers are connected to each other via an internet. These number of server are communicate with each other that is inter server communication. All clients are communicating to each other with the help of server. On the availability of the server client is assigned to that server. All nodes that is computers are connected to each other to share resources and computation. To communication within inter server or clients there is interlatency and load balancing problem. If one server fail then automatically that server load is assigned to next server. This increases performance problem. Due to resuming load to another server load is increased on the network. For better performance one has to manage that load. There are so many clustering algorithms to make an partition a group into object. But no clustering algorithm has knowledge of minimizing total communication cost.

Keyterms: *Load Balance, client assignment Problem, round robin, Load distance balancing problem, Round robin etc.*

*****

I. Introduction

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 communication load and 2) Load balancing of the servers. In our approach we propose an algorithm that is based on Pre-emptive scheduling method to obtain an approximately optimal solution for the client-server assignment problem.

1. The two groups have different number of servers, a server within a group with fewer servers will likely to have a higher load than a server in the group with more servers. This reduces the load balance
2. We describe a number of emerging applications that have the potential to benefit from the client-server assignment problem.

The modern internet is a collection of interconnected networks of various systems that share resources. A superlative 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 overhead of coordinating nodes, which results in lower performance. Typical distributed system consists of servers and clients. Servers are more computational and provide powerful resources than clients. Examples of such systems are 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 communication load and load balancing on servers. If one server is overloaded, we need to add another server to distribute the load, which is economically inefficient and usually increases the overall communication load As a heavily loaded server typically exhibits a low performance, we would like to avoid the situation. To minimize the amount of total communication load, assigning all clients to one server is optimal. However, it is impossible due to overloading and completely loses the load balance. Simple load balancing does not usually take account of reducing the overall communication load.any interaction between two clients consists of both clients to server latency and inter-server latency which is called an interaction path.

II. Related Work

Load-Distance balancing Problem Scheduling is the method by which threads, processes or data flows are given access to system resources (e.g. processor time, communications bandwidth). This is usually done to load balance and share system resources effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (executing more than one process at a time) and multiplexing (transmit multiple data streams simultaneously across a single physical channel).

The scheduler is concerned mainly with:

- Throughput - The total number of processes that complete their execution per time unit.
- Latency, specifically: Turnaround time - total time between submission of a process and its completion. Response - amount of time it takes from when a request was submitted until the first response is produced.
- Fairness - Equal CPU time to each process (or more generally appropriate times according to each process' priority and workload).
- Waiting Time - The time the process remains in the ready queue.

In practice, these goals often conflict (e.g. throughput versus latency), thus a scheduler will implement a suitable compromise. Preference is given to any one of the above mentioned concerns depending upon the user's needs and objectives.

System Architecture

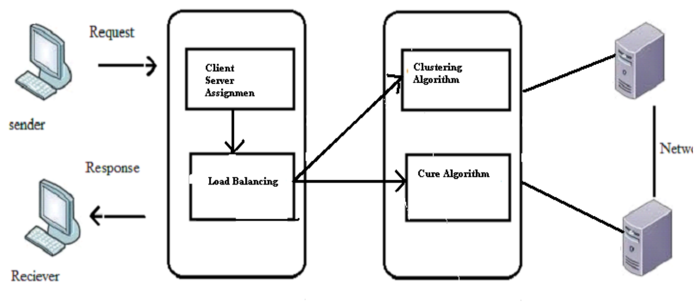

Fig.1 Client server assignment in Networking

Clustering Algorithm

```
1.  Input: Let {T1, T2,…,Tk} be the accepted tasks in the ready queue and let $e_i$ be the expected execution time of $T_i$. Let current time be t and let $T_0$ be the task currently being executed. Let the expected utility density threshold be μ.
2.  If a new task, i.e., $T_p$ arrives then
3.  Check if $T_p$ should pre-empt the current task or not;
4.  If Pre-emption allowed then
5.  $T_p$pre-empts the current task and starts being executed;
6.  End if
7.  If Pre-emption not allowed then
8.  Accept $T_p$ if $U_p(e_0)$  > μe;
                                 $e_0$
9.  Reject $T_p$ if $U_p(e_0)$  >μ$e_{0;}$
                                 $e_0$      $e_0$
10. end if
11. Remove $T_j$ in the ready queue if $U_p(e_0)$  > μ;
        $e_j$
12. End if
13. If at pre-emption check point then
14. PREEMPTION CHECKING;
15. End if
16. If $T_0$ is completed then
17. Choose the highest expected utility density task $t_i$ to run.
18. Remove $T_j$ in the ready queue if $U_j(e_i)$   > μ;
        $e_j$
19. End if
20. If t= the critical time of τ0 then
21. Abort τ0 immediately;
22. Choose the highest expected utility density task τi to run.

23. Remove τj in the ready queue if  Uj(Cj) ≤ δ;
Cj
24. End if;
```

Cure Algorithm

CURE (Clustering Using Representatives) is an efficient data clustering algorithm for large databases that is more robust to outliers and identifies clusters having non-spherical shapes and wide variances in size.To avoid the problems with non-uniform sized or shaped clusters, CURE employs a novel hierarchical clustering algorithm that adopts a middle ground between the centroid based and all point extremes. In CURE, a constant number c of well scattered points of a cluster are chosen and they are shrunk towards the centroid of the cluster by a fraction α. The scattered points after shrinking are used as representatives of the cluster. The clusters with the closest pair of representatives are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. This enables CURE to correctly identify the clusters and makes it less sensitive to outliers
The algorithm is given below.

The running time of the algorithm is O(n^2 log n) and space complexity is O(n).The algorithm cannot be directly applied to large databases. So for this purpose we do the following enhancements

- Random sampling : To handle large data sets, we do random sampling and draw a sample data set. Generally the random sample fits in main memory. Also because of the random sampling there is a tradeoff between accuracy and efficiency.
- Partitioning for speed up : The basic idea is to partition the sample space into p partitions. Each partition contains n/p elements. Then in the first pass partially cluster each partition until the final number of clusters reduces to n/pq for some constant $q \geq 1$. Then run a second clustering pass on n/q partial clusters for all the partitions. For the second pass we only store the representative points since the merge procedure only requires representative points of previous clusters before computing the new representative points for the merged cluster. The advantage of partitioning the input is that we can reduce the execution times.
- Labeling data on disk : Since we only have representative points for k clusters, the remaining data points should also be assigned to the clusters. For this a fraction of randomly selected representative points for each of the k clusters is chosen and data point is assigned to the cluster containing the representative point closest to it.

Steps of algorithm

CURE(no. of points)

Input : A set of points S

Output :k clusters

1. For every cluster u (each input point), in u.mean and u.rep store the mean of the points in the cluster and a set of c representative points of the cluster (initially c = 1 since each cluster has one data point). Also u.closest stores the cluster closest to u.
2. All the input points are inserted into a k-d tree T
3. Treat each input point as separate cluster, compute u.closest for each u and then insert each cluster into the heap Q. (clusters are arranged in increasing order of distances between u and u.closest).
4. While size(Q) >k
5. Remove the top element of Q(say u) and merge it with its closest cluster u.closest(say v) and compute the new representative points for the merged cluster w.
6. Also remove u and v from T and Q.
7. Also for all the clusters x in Q, update x.closest and relocate x
8. insert w into Q
9. repeat

Comparative Analysis of Algorithms

Parameter	**Clustering Algorithm**	**Cure Algorithm**
Start Time	155718 ms	155850 ms
End Time	155727 ms	155852 ms
Total Execution	9 ms	2 ms

Table 1: Analysis of Algorithm

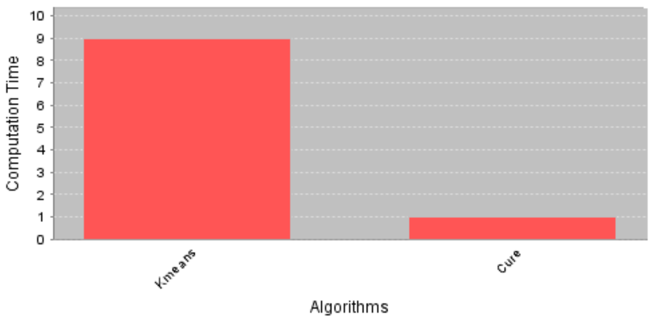

Fig.2 Comparative Study of Clustering and CURE Algorithm

III. Architectural Design

A typical distributed system consists of a mix of servers and clients. The servers are more computational and resource powerful than the clients. A classic example of such systems is e-mail. When a client A sends an e-mail to another client B, A does not send the e-mail directly to B. Instead, A sends its message to its e-mail server which has been previously assigned to handle all the e-mails to and from A. This server relays A’s e-mail to another server which has been previously assigned to handle e-mails for B. B then reads A’s e-mail by downloading the e-mail from its server. Mainly, the e-mail servers communicate with each other on behalf of their clients[1]. The main advantage of this architecture is that the powerful dedicated e-mail servers release their clients from the responsibility associated with many tasks including processing and storing e-mails, and thus making e-mail applications more scalable. A more interesting scenario is the Instant Messaging System (IMS). An IMS allows real time text-based communication between two or more participants over the Internet. Each IMS client is associated with an IMS server which handles all the instant messages for its clients. Similar to e-mail servers, IMS servers relay instant messages to each other on behalf on their clients. In an IMS that uses the XMPP (Jabber) protocol such as Google Talk, clients can be assigned to servers independent of their organizations.

Problems on existing system:

1. If one server is overloaded, we need to add another server to distribute the load, which is economically inefficient and usually increases the overall communication load
2. As a heavily loaded server typically exhibits a low performance, we would like to avoid the situation.
3. To minimize the amount of total communication load, assigning all clients to one server is optimal. However, it is impossible due to overloading and completely loses the load balance. Simple load balancing does not usually take account of reducing the overall communication load.

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 communication load and 2) Load balancing of the servers. In our approach we propose an algorithm that is based on Semi definite programming, to obtain an approximately optimal solution for the client-server assignment problem.

1. The two groups have different number of servers, a server within a group with fewer servers will likely to have a higher load than a server in the group with more servers. This reduces the load balance
2. We describe a number of emerging applications that have the potential to benefit from the client-server assignment problem.

IV. CONCLUSION

To make a partition of an object for distributed system clustering algorithms are used to make an partition. But it fails in throughput. Round robin algorithm does not show current status of the system. But pre-emptive algorithm work on scheduling of process of client with respective to time and performance. As compare to clustering algorithm CURE algorithm has total execution cost is very low.

References

[1] Hiroshi Nishida, Member, IEEE, and Thinh Nguyen, Member, IEEE-“Optimal Client-Server Assignment for Internet Distributed Systems”-ieee transactions on parallel and distributed systems, vol. 24, no.3, march 2013.

[2] “Finding good nearly balanced cuts in power law graphs,” YahooResearch Labs, Tech. Rep., 2004.

[3] Nishida, H.; Thinh Nguyen; , "Optimal Client-Server Assignment forInternet Distributed Systems," Computer Communications andNetworks (ICCCN), 2011 Proceedings of 20th International Conferenceon , vol., no., pp.1-6, July 31 2011-Aug. 4 2011

[4] Lu Zhang; Xueyan Tang; , "Client assignment for improvinginteractivity in distributed interactive applications," INFOCOM, 2011Proceedings IEEE , vol., no., pp.3227-3235, 10-15 April 2011doi: 10.1109/INFCOM.2011.5935173

[5] Zhang, L.; Tang, X.; , "Optimizing Client Assignment for EnhancingInteractivity in Distributed Interactive Applications," Networking,IEEE/ACM Transactions on , vol.PP, no.99, pp.1, 0doi: 10.1109/TNET.2012.2187674

[6] Bortnikov, E., Khuller, S., Li, J., Mansour, Y. and Naor, J. S. (2012),The load-distance balancing problem. Networks, 59: 22–29. doi:10.1002/net.20477

[7] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and VenkataramananBalakrishnan. Linear Matrix Inequalities in System and Control Theory.SIAM, 1994.

[8] U. Feige and M. Langberg. The rpr2 rounding technique forSemidefinite programs. J. Algorithms, 60(1):1–23, 2006.

[9] B. Scho¨lkopf, A. Smola, and K.-R. Mu¨ ller, “Nonlinear ComponentAnalysis as a Kernel Eigenvalue Problem,” Neural Computation,vol. 10, pp. 1299-1319, July 1998.

[10] G. Karypis and V. Kumar, “A Fast and High Quality MultilevelScheme for Partitioning Irregular Graphs,” SIAM J. ScientificComputing, vol. 20, pp. 359-392, Dec. 1998.

[11] M.L. Huang and Q.V. Nguyen, “A Fast Algorithm for BalancedGraph Clustering,” Proc. 11th Int’l Conf. Information Visualization,pp. 46-52, 2007.

[12] K. Andreev and H. Ra¨cke, “Balanced Graph Partitioning,” Proc.16th Ann. ACM Symp.Parallelism in Algorithms and Architectures,pp. 120-124, 2004.

[13] F. Nie, C. Ding, D. Luo, and H. Huang, “Improved MinMaxCutGraph Clustering with Nonnegative Relaxation,” Proc. EuropeanConf. Machine Learning and Knowledge Discovery in Databases: PartII, pp. 451-466, 2010.

[14] P. Chan, M. Schlag, and J. Zien, “Spectral K-Way Ratio-CutPartitioning and Clustering,” IEEE Trans. Computer-Aided Design ofIntegrated Circuits and Systems, vol. 13, no. 9, pp. 1088-1096, Sept.1994.

[15] C.H.Q. Ding, X. He, H. Zha, M. Gu, and H.D. Simon, "A Min-MaxCut Algorithm for Graph Partitioning and Data Clustering," Proc.Int'l Conf. Data Mining (ICDM '01), pp. 107-114, 2001.

[16] H.S. Stone, "Multiprocessor Scheduling with the Aid of NetworkFlow Algorithms," IEEE Trans. Software Eng., vol. 3, no. 1, pp. 85-93, Jan. 1977.

[17] P. Sinha, Distributed Operating Systems: Concepts and Design. IEEEPress, 1997.

[18] J.C.S. Lui and M.F. Chan, "An Efficient Partitioning Algorithmfor Distributed Virtual Environment Systems," IEEE Trans.Parallel and Distributed Systems, vol. 13, no. 3, pp. 193-211, Mar.