International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:1, 2012

A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems

Ali M. Alakeel

***Abstract*—**Load balancing in distributed computer systems is the process of redistributing the work load among processors in the system to improve system performance. Most of previous research in using fuzzy logic for the purpose of load balancing has only concentrated in utilizing fuzzy logic concepts in describing processors load and tasks execution length. The responsibility of the fuzzy-based load balancing process itself, however, has not been discussed and in most reported work is assumed to be performed in a distributed fashion by all nodes in the network. This paper proposes a new fuzzy dynamic load balancing algorithm for homogenous distributed systems. The proposed algorithm utilizes fuzzy logic in dealing with inaccurate load information, making load distribution decisions, and maintaining overall system stability. In terms of control, we propose a new approach that specifies how, when, and by which node the load balancing is implemented. Our approach is called Centralized-But-Distributed (CBD).

***Keywords*—**Dynamic load balancing, fuzzy logic, distributed systems, algorithm.

I. Introduction

LOAD balancing in distributed computer systems is the process of redistributing the work load among processors in the system to improve system performance [1]. Various studies have shown that distributing the work load evenly among nodes of a distributed system highly improves system performance and increases resource utilization. Dynamic load balancing algorithms monitor changes on the system work load and redistribute the work load accordingly, e.g., [1]-[13]. A dynamic load balancing algorithm is usually composed of three strategies: transfer strategy, location strategy, and information strategy. Transfer strategy decides on which tasks are eligible for transfer to other nodes for processing. Location strategy nominates a remote node to execute a transferred task. Information strategy is the information center of a load balancing algorithm. It is responsible for providing location and transfer strategies at each node with the necessary information required to take their decisions. Information strategy is an important part of a load balancing algorithm. The worth and complexity of any dynamic load balancing algorithm depends heavily on the performance of its information strategy. The implementation responsibility or control of a dynamic load balancing algorithm can take three different forms: centralized, distributed, or semi-distributed.

Ali M. Alakeel is with the College of Computers and Information Technology, University of Tabuk, P.O.Box 741, Tabuk 71491, Saudi Arabia (e-mail: alakeel@ut.edu.sa).

In a centralized load distribution algorithm, a single node (called central node) in the network is nominated to be responsible for all load distribution in the network. In a distributed load balancing algorithm, the responsibility is distributed where each node in the network carries an equal share of the responsibility and executes the same algorithm. In a semi-distributed load balancing algorithm, the network is segmented into clusters where each cluster contains a set of nodes. The control within each cluster is centralized, i.e. a central node is nominated to take charge of load balancing within its set. Load balancing of the whole distributed system is achieved through the cooperation of central nodes of each cluster, i.e. the responsibility is distributed among the central nodes of each cluster.

Most of previous research in using fuzzy logic for the purpose of load balancing, e.g., [16]-[28], has only concentrated in utilizing fuzzy logic concepts in describing processors load and tasks execution length. The responsibility of the fuzzy-based load balancing process itself, however, has not been discussed and in most reported work is assumed to be performed in a distributed fashion by all nodes in the network. This distributed approach requires that the load balancing algorithm resides in all nodes of the network all of the time. Because of the intrinsic difference between rule-based applications and functional based application, using a distributed approach for the load balancing control mechanism may not be the best choice and may increase the costs of the load balancing process. This is because rule-based applications require more memory and procession power resources than functional based applications. The memory requirement is associated with the knowledge base and during the rules chaining process, while the processing power is required by the inference engine during the search process for a given solution or a goal. Moreover, the distributed approach for the load balancing process still has some traditional problems, one of which is that optimal scheduling decisions are difficult to make because of the rapidly changing environment introduced by the arrivals and departures from individual processors. Another disadvantage is the extra communication overhead introduced by all processors trying to gather information about each other.

This paper proposes a new fuzzy dynamic load balancing algorithm for homogenous distributed systems. The proposed algorithm utilizes fuzzy logic, e.g., [29]-[31], in dealing with inaccurate load information, making load distribution decisions, and maintaining overall system stability. In terms of control, we propose a new approach that specifies how, when,

and by which node the load balancing is implemented. Our approach is called Centralized-But-Distributed (CBD). Also we introduce a new location policy which utilizes Fuzzy Logic techniques in redistributing the load from heavily loaded nodes to lightly loaded nodes in the system. Our load balancing algorithm includes an explicit mechanism for monitoring and tuning system stability. System stability is maintained through the dynamic adjustment and tuning of various parameters incorporated in the algorithm.

The rest of this paper is organized as follows. Section II provides an overview of different approaches to control the process of load balancing in a distributed system environment. Section III presents our proposed fuzzy dynamic load balancing algorithm and in Section IV we discuss our conclusions and future research.

II. Responsibility Of Load Balancing

Along with various load balancing strategies which may be applied independently or tailored to enhance the performance of an algorithm for solving a certain problem, different policies of where to put the control of the load balancing algorithm have been proposed in the literature: centralized, distributed, or semi-distributed.

A centralized load balancing strategy assigns a single processor the responsibility of initiating and monitoring the load balance operation. In this strategy, a dedicated processor gathers the global information about the state of the system and assigns tasks to individual processors. Despite its high potential of achieving optimal performance, centralized strategies have some disadvantages: high vulnerability to failures, storage requirements for maintaining the state information - especially for large systems, and the dependability of the performance of the system on the central processor which could result in a bottleneck [1].

In a distributed load balancing strategy, each processor executes the same algorithm and exchanges information with other processors about the state of the system. Each processor may send or receive work on the basis of a sender-initiated or a receiver-initiated policy. In a sender-initiated policy, the sender decides which job gets sent to which receiver. In a receiver-initiated policy, the receiver searches for more work to do. Intuitively, queues are formed at senders if a receiver-initiative policy is used, while they are formed at receivers if a sender-initiative policy is used. Additionally, scheduling decisions are made when a new job arrives at the sender in a sender-initiative, while they are made at the departure of a job in a receiver-initiative policy. The determination of which policy is adopted depends upon the load transfer request which can be initiated by an over-loaded or under-loaded processor [1], [3], [6], [7]. It has been demonstrated in [4], [6], and [8], using analytical models and simulations, that sender-initiated strategies generally perform better at lower system loads while receiver-initiated strategies perform better at higher system loads, assuming that process migration cost under the two strategies is comparable. Some of the advantages offered by the distributed policy are: Fault tolerance, minimum storage requirements to keep status information, and the availability of system state information at all nodes. The distributed policy still has some disadvantage, one of which is that optimal scheduling decisions are difficult to make because of the rapidly changing environment introduced by the arrivals and departures from individual processors. Another disadvantage is the extra communication overhead is introduced by all processors trying to gather information about each other. To mitigate this overhead, some distributed strategies minimize the amount of information exchanged, which has a negative reflection on the performance of an algorithm.

The semi-distributed policy comes in the middle between centralized and distributed policies. It is introduced to take the best of each and to avoid the major drawbacks of each of the two policies. The semi-distributed strategy is based on the partitioning of the processors into equal sized sets. Each set adopts a centralized policy where a central processor takes charge of load balancing within its set. The sets together adopt a distributed policy where each central processor of each set exchanges information with other central processors of other sets to achieve a global load balance.

It has been shown in [1] that the semi-distributed policy produces a better performance than the centralized and distributed policies. Research demonstrates that each central processor yields optimal load balance locally within its set. Moreover, this policy does not incur high communication overhead while gathering system state information. Although this policy is a mediator between the centralized and the distributed ones, it fits large distributed systems better than small systems.

III. The Proposed Fuzzy Load Balancing Algorithm

A. System Model

In this presentation we assume the following system model. We assume N, where $N > 1$, independent homogenous nodes are connected by a local area network where each node consisting of a signal processor. A single typed tasks arrive to a node could either come from outside the network or from other nodes in the network. We assume that all nodes are subjected to the same average arrival rate of tasks coming from outside the network. All tasks are queued at each node and are served on a First Come First Serve (FCFS) basis.

B. Assumptions

- Nodes of the distributed system are numbered from 1 to N, where N is the total number of nodes in the system. The numerical number of each node represents its identification (ID) in the system.
- Nodes are connected by a broadcast network and the cost of sending a message between any two nodes is the same.
- We assume that each processor is in some state S_k where it has a number of tasks T_k. Furthermore, we assume that the distributed system is initially in a steady state.
- Given this configuration, the load balancer starts and tries to examine the overall state of the system and takes the necessary corrective actions accordingly based on the objectives this algorithm assumes.

C. Objectives

Once a node is elected to act as the load balancer of the distributed system it has the following objectives:

1) Obtains full information about the load of each host in the system efficiently.
2) Maintains a load balance among the distributed system hosts within a difference *d*. The value of *d* varies during the operation of the load balancing algorithm and adjusts dynamically taking into account the current state of the system and the communication costs. The proper range of *d* will only be determined after experimentations with this algorithm because this process has to be done efficiently in terms of communication time required.
3) Selects the most appropriate time to launch the load balancing process. Since keeping the load balancer working at all times is a burden on the system, we propose an efficient way of triggering the load balancer, which is described as follows.

The load balancer algorithm is triggered when one of the following conditions is satisfied:

- When a node becomes idle, or below a threshold value.
- When the load of a node exceeds a threshold value.

4) Ensures that only one node is working as the load balancer at a time. It is possible that more than one node can meet either one of the above conditions. This would cause multiple load balancers to be active at the same time. To prevent this, our algorithm provides a mechanism which ensures that only one load balancer is active at a given time.

D.The Algorithm Steps

The load balancer node performs the following steps:

1) Obtain the current load of the distributed system. This is accomplished by broadcasting a request for status message to all nodes in the system.
2) Upon receiving the response from all nodes, the load balancer assigns each node of the system including itself a fuzzy value in the interval [0,1] which represents the load of that node relative to the overall load of the distributed system.

This assignment is achieved by forming a fuzzy set, LOADED = {light, normal, heavy}, that represents the load of the system. Using Fuzzy Logic techniques each node of the distributed system is assigned a membership value depending on its current load. The membership value is in the interval [0,1] and reflects the compatibility of the load at a specific node to the fuzzy term LOADED which is represented by a fuzzy set. The assignment of membership values (grades) is based on the S-function [31] which is shown in Fig. 1.

3) Using the results of step (2), the load balancer categorizes all nodes into three separate groups: Underloaded, Normal, and Overloaded. Where Normal nodes do not need any load balancing and will be left unchanged. Both Underloaded and Overloaded nodes need help and will be the target of the load balancing process.

$$S(x;\ \alpha,\beta,\gamma) = \begin{cases} 0 & for\ x \le \alpha \\ 2\ \left(\frac{x-\alpha}{\gamma-\alpha}\right)^2 & for\ \alpha \le x \le \beta \\ 1-2\ \left(\frac{x-\gamma}{\gamma-\alpha}\right)^2 & for\ \beta \le x \le \gamma \\ 1 & for\ x \ge \gamma \end{cases}$$

Fig. 1 The S-function

4) Create a mapping from Overloaded nodes to Underloaded nodes. The outcome of this mapping tells each overloaded node where to ship some of the extra work it has. As a result of this mapping, the load balancer sends each overloaded node a message specifying the ID of each possible underloaded node and the number of tasks the overloaded node should ship to the underloaded node. This information is formed in a list arranged as: (ID_1, #tasks), (ID_2, #tasks), (ID_N, #tasks). To perform this mapping, we adopt Evans et al. [31] probability model by using the load at each node and compute the probability of sending a task from an overloaded node *i* to and underloaded node *j*.

The main contribution of our proposed algorithm is in the part which is responsible for selecting the best time to trigger the load balancing process and in performing the load balancing process itself. In this regard, a copy of the load balancing algorithm resides on each host as in the distributed load balancing algorithm reported in the literature. In our approach, the responsibility of load balancing is centralized in action, but distributed in time. This means that only a single node will be responsible for performing the load balancing processes of the distributed system. This node, however, is not the only one who will complete this task all of the time, as is in the traditional centralized approach. Instead, each node might have the chance to perform the load balancing task. The selection of which node actually does the work is dynamically determined, however. By doing this, our approach attempts to get the best of the well-know centralized and distributed approaches to the load balancing problem.

IV. Conclusion

This paper propose a new dynamic load balancing algorithm for homogenous distributed computer systems which employs fuzzy logic in dealing with inaccurate load information, making load distribution decisions, and maintaining overall system stability. The main contribution of our proposal is in the part which is responsible for selecting the best time to trigger the load balancing process and in performing the load balancing process itself. In order to evaluate the proposed algorithm, we intend to perform and extensive experimental study using simulation to show the effectiveness of the proposed algorithm as compared to existing dynamic load balancing algorithms reported in the literature.

REFERENCES

[1] I. Ahmed and A. Ghafoor, "Semi-Distributed Load Balancing for Massively Parallel Multicomputers," IEEE Trans. Software Eng., vol. 17, no. 10, pp 987-1004, October 1991.
[2] T. L. Casavant, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," IEEE Trans. Software Eng., vol 14, no. 2, pp 141-154, February 1988.
[3] Y. Wang and R. Morris, "Load Sharing in Distributed Systems," IEEE Trans. Comput., vol. C-34, no. 3, pp. 204-217, Mar. 1985.
[4] K. Ramamritham, J. A. Stankovic, and W. Zhao, "Distributed Scheduling of Tasks with Deadlines and Resource Requirements," IEEE Trans. Comput., vol. 38, no. 8, pp 1110-1123, August 1989.
[5] J. A. Stankovic, K. Ramamritham, and S. Cheng, "Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems," IEEE Trans. Comput., vol. C-34, no. 12, pp. 1130-1143, December 1985.
[6] D.L. Eager, E.D. Lazowski, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," IEEE Trans. Software Eng., vol. SE-12, no. 5, pp. 662-675, May 1986.
[7] L. M. Ni, C. Xu, and T. B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing," IEEE Trans. Software Eng., vol.SE-11, no. 10, pp. 1153-1161, October 1985.
[8] D. L. Eager and E. D. Lazowski, and J. Zahorjan, "A Comparision of Receiver-Inititated and Sender Initiated Adaptive Load Sharing," Performance Evaluation, 6, pp. 53-68, March, 1986.
[9] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," J. Parallel Distrib. Comput. 7 (1989), pp. 279–301.
[10] J. Watts, S. Taylor, "A practical approach to dynamic load balancing," IEEE Trans. Parallel Distrib. Systems 9 (3) (March 1998), pp. 235–248.
[11] P. Krueger, N.G. Shivaratri, "Adaptive location policies for global scheduling," IEEE Trans. Software Eng. 20 (6), pp. 432-444, June 1994.
[12] S. Dhakal, M. M. Hayat, J.E.Pezoa, C. Yang, and D. Bader, "Dyanmic Load Balancing in Distributed System in the Presence of Delays: A Regeneration-Therory Approach,", IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, April 2007.
[13] D. J. Evans and W.U.N. Butt," Dynamic load balancing using task-transfer probabilities," Parallel Computing, Vol. 19, No. 8, pp. 897-916, August 1993.
[14] A. M. Alakeel, "Load Balancing in Distributed Computer Systems," Int. Journal of Computer Science and Information Security, Vol. 8, No. 4, pp. 8-13, July 2010.
[15] A. M. Alakeel, "A Guide to Load Balancing in Distributed Computer Systems," Int. Journal of Computer Science and Network Security, Vol. 10, No. 6, pp. 153-160, June 2010.
[16] K. Abini, "Fuzzy Decision Making for Load Balancing in a Distributed System, " Proceedings of the 36th Midwest Symposium Circuits and Systems, pp. 500–502, 1993.
[17] Yu-Kwong Kwok, Lap-Sun Cheung, "A new fuzzy-decision based load balancing system for distributed object computing," Journal of Parallel and Distributed Computing, Volume 64, Issue 2, pp. 238-253, February 2004.
[18] L. Singh, A. Narayan, and S. Kumar, "Dynamic fuzzy load balancing on LAM/MPI clusters with applications in parallel master-slave implementations of an evolutionary neuro-fuzzy learning system," IEEE International Conference on Fuzzy Systems, pp.1782-1788, , June 2008.
[19] C.W. Cheong, V. Ramachandran, "Genetic Based Web Cluster Dynamic Load Balancing in Fuzzy Environment," Proceedings of the Fourth International Conference on High Performance Computing in the Asia-Pacific Region, Beijing, China, Vol. 2, pp. 714–719, 2000.
[20] P. Chulhye, J.G. Kuhl, "A fuzzy-based distributed load balancing algorithm for large distributed systems," Proceedings of the Second International Symposium on Autonomous Decentralized Systems, pp. 266–273, April 1995.
[21] M. Rantonen, T. Frantti, and K. Leiviskä, "Fuzzy expert system for load balancing in symmetric multiprocessor systems," Journal of Expert Systems with Applications, Vol. 37 No. 12, pp. 8711-8720, December, 2010.
[22] L. Singh, A. Narayan, and S. Kumar, "Dynamic fuzzy load balancing on LAM/MPI clusters with applications in parallel master-slave implementations of an evolutionary neuro-fuzzy learning system," IEEE International Conference on Fuzzy Systems, pp.1782-1788, June 2008.
[23] I. Barazandeh, S. S. Mortazavi, and A. M. Rahmani, "Intelligent fuzzy based biasing load balancing algorithm in distributed systems," IEEE 9th Malaysia International Conference, pp.713-718, Dec. 2009.
[24] E. El-Abd, "Load blancing in distrubuted computing systesms using fuzzy sexpert ssytems, " Int. Confernce on Modern Probmesn of Radio Engiennering, Telecommunications and Computer Scinece, Lviv-Slavsko, Ukraine, pp. 141-144, 2000.
[25] S. Dierkes, "Load balancing with a fuzzy-decision algorithm," Inform. Sci. 97 (1–2), pp. 159–177, March 1997.
[26] Shaout, P. McAuliffe, "Job scheduling using fuzzy load balancing in distributed system," Electron. Lett. 34 (20), pp. 1983–1985, October 1998.
[27] K.-W.Wong, "Fuzzy routing control of service request messages in an individual computing environment," Proceedings of ACM Symposium on Applied Computing, Nashville, TN, pp. 548–551, 1995.
[28] Ajoy Kumar, Mukesh Singhal, Ming T(M&e) Liu, "A Model for Distributed Decision Making: ***An*** Expert System for Load Balancing in Distributed Systems", IEEE computer software and applications conference, 1987.
[29] L. A. Zadeh, "Fuzzy Sets," Information and Control, No. 8, pp. 338-353, 1965.
[30] B. Kosko, "Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence," Prentice-Hall, Englewood Cliffs, NJ, 1992.
[31] J. Giarratano, "*Expert Systems: Principles and Programming*," PWS-KENT Publishing Company, Boston, 1989.