Fuzzy based Design and tuning of distributed systems load balancing controller

Seyed Rasool Moosavi-Nejad, Islamic Azad University, Arak Branch,s.r.mosavi@gmail.com
S.S. Mortazavi, ,Shahid Chamran University
mortazavi_s@scu.ac.ir
Bijan Vosoughi Vahdat, , Sharif University of Technology
vahdat@sharif.edu
Name of the Presenter: Seyed Rasool Moosavi-Nejad

Abstract

The load model of the distributed systems tends to change dynamically. Hence, fast. Precise Load balancing is an essential factor in increasing distributed system operation efficiency. Lack of shared memory among independent systems, and the delays in interface channels, leads to ambiguous information of the condition of system. This ambiguity brings about uncertainty in decision for load balancing. To solve this problem; this paper proposes an intelligent algorithm based on fuzzy logic in the centralized distributed system. In comparative study, the length of sent packets and the service rate in each node are considered variable. The fuzzy decision is done based on the current load and waiting time in this paper. Throughput, average of response time, and drop rate of the packets are considered as measures in a comparative study with other static and dynamic algorithms.

Key words: Distributed systems, fuzzy controller, load balancing

1. Introduction

Due to increasing development of computing technology there has been an increase in computer's efficiency and a decrease in their volumes. As the small powerful computers have been developed, the distributed processing has been introduced as an optimum solution to achieve great processing power using existing resources. In this method, each task is divided in to a series of small task, and forwarded to several processors. Inherently distributed applied programs, share data among distributed users. Sharing sources result in better efficiency to cost ratio, less response time, higher throughput, more reliability, extensibility and double development and better flexibility in observation. The increasing need of computing capacity has caused the computing environment to change from centralized to distributed state. Hence, distributed systems play a major role in computing requirements. Since network and computing resources are restricted, load balancing is essential in distributed systems in order to achieve higher productivity, availability, and stability. In general, load balancing algorithm aims to distribute the load on resources properly to maximize their efficiency while minimizing overall execution time of the tasks [17]. In other words, an ideal load balancing is an algorithm that eventually enables all the nodes to complete their tasks concurrently [16]. The methods suggested for load distribution are divided into two very general categories: a) algorithms which aim to equalize the processing load among all the nodes. b) Algorithms which aim to prevent the idleness of some node while there are requests queuing in other nodes. In other words, these algorithms are not intended to completely equalize the load of nodes [9]. Load balancing methods are divided into other categories such as central and non-central, static and dynamic, periodic and aperiodic, with and without threshold limit [1, 10, 11, 15]. In central algorithms, there is a central node that collects the data of the load of all nodes and makes a suitable decision for load distribution. These algorithms are not reliable as they are dependent on one node. Any failure of the central node results in the failure of the load balancing procedure. However, they are very simple in implementation, and have less communication overload than non-central methods. In non-central models there is hardly one or more specific node, i.e. server or collector but all the nodes contain data of all or part of other nodes. Therefore, each node can decide on load balancing, all by itself. in the distributed approach, there is problem of the information about the nodes becoming obsolete and there is communication overload. Another important classification for load balancing algorithm, is dividing them into static and dynamic groups. Static algorithms are not affected by system condition and their behavior is pre-defined. Random node selection and round robin node selection are among such algorithms [3]. A major problem with static method is that they are not adaptive to the change in the general state of the system. Dynamic algorithms make their decisions according to the current condition of the system. The efficiency of these algorithms is dependent on the selection of the criterion for estimation of the load rate. On the whole, these methods have better efficiency and scalability than that of static methods [4]. This better efficiency results in complicated algorithm architecture and increase overload in collecting data from the system.

Another approach is to use numerical analysis and probabilities to predict the situation of the system for load balancing [5]. However, using numerical analysis needs very complicated computations. Furthermore, complexity may cause instability in distributed systems.

From another point of view, regarding which of the nodes sends the request for load balancing, classic load balancing algorithms are divided into three categories: 1) sender-initiated, 2) receiver-initiated and 3) a combination of the both [12]. The existence of large communication overload and increased load in the overloaded nodes is among the problems of these kinds of algorithms.

A good load balancing algorithms must be fast and should not add heavy load to the nodes, while having a correct view on the state of the whole system.

Load model of the distributed systems tend to change dynamically and very fast. Lack of shared memory among independent system and the delay in communication channels incurs ambiguous information about the whole system situation. This ambiguity causes uncertainty in decision-making. To solve this problem, artificial intelligence and intelligent algorithms such as fuzzy logic [8] and genetic algorithm [18] are suggested to use. In [13], a fuzzy approach is provided in which the ambiguity of the system situation is being considered, and enjoys a better efficiency than that of classic algorithm. The space between the nodes and the rate of the load of the system, are the input parameters of this algorithm. The load of servers and the response time are two main parameters for selecting servers that are shown to do better than classic methods. In [6] a fuzzy base approach is used for load balancing in computer network. Paper [6] uses the two method of sender-initiated and receiver-initiated offering a new algorithm by changing the membership function in the fuzzy state. The problem is that determining and setting fuzzy controller components including membership function is not an easy, handy task and the offered controllers are not often optimum. In that paper, Mr. Lee has provided a genetic algorithm-based method for sender-initiated load balancing in distributed system and has designed a proper merit function. In that plan, the processors to which the requests are to be forwarded are determined by genetic algorithms. Unnecessary request are consequently reduced. The major disadvantage with this approach is that it is time-consuming, preventing it to be ideal for concurrent application. This paper presents a new fuzzy-base intelligent load balancing algorithm. The effectiveness of the proposed algorithm will be compared with that of dynamic and static algorithms.

2. System model:

Figure 1 shows the architecture of distributed system that is used as test bed in the present paper. This system architecture is hierarchal and the central method is applied for load balancing. In this way, communication overload resulting from data collection is considerable decreased.

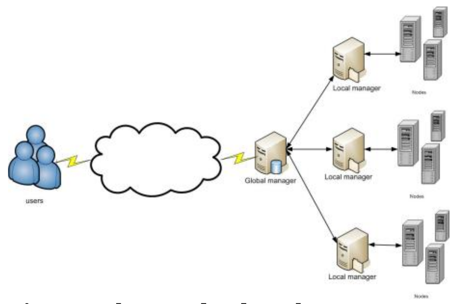

Fig. 1: The applied architecture

In the designed architecture, the load balancing task is performed at two levels: i) group level by global manager and ii) local manager at node level. Several nodes constitute one group. In each group, there is a node which functions as designated representative that performs local load balancing task in its own group and communicates with group manager. This paper considers another node to maximize fault-tolerance. The nodes of each group only communicate with the group manager. The task allocated to each group or node, will be processed in the same group and node and will not be relocated. Groups will not communicate with each other. Global manager communicates with the manager of each group.

3. The proposed algorithm

In this algorithm, the weights of assignment for the nodes and the groups are decided. This weight assignment is carried out in two levels of global and local managers. Base on the generated weight, a percentage of tasks will be send to the local group by global manager. Then, in the local group, based on the weight of the nodes, a percentage of the transferred task will be send to the nodes. Assignment of weight is made by fuzzy controller. The criterion for decision-making is bases on two variables of the current load and waiting time of the last processed task. Figure 2 shows fuzzy controller structure

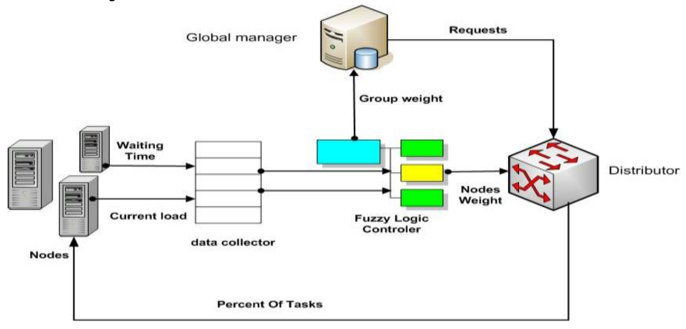

Fig 2: Components of fuzzy controller and the exchange of data between them.

The process of load balancing is handled in 3 stages:

3.1 The stage of data collection

Fuzzy controller needs data to generate weights. Each node sends its state information for its group manager that carries out the local balancing, in a specifying time which is the time for specifying the state. Each local manager does the task of generating weight, base on the data received from the nodes of its group. Based on this data, fuzzy controller also assigns total weight of the group and sends it for global manager. Based on the data received from group managers, the global manager decides and distributes the load among the groups

To exchange the data, each of the nodes notifies the manager about its load state periodically. In other words, to specify the current state of the groups and nodes, the balancing data in the system, is updated without any request. The manager of each group sends the global state of its group to the global manager, based on received data from the nodes. By this method, many redundant and extra requests are dropped from the system. Hence, the system overload is greatly decreased. The most important issue in this method is determining a proper time period to exchange data. Very long intervals for exchanging data, cause inexact decision-making, while short time intervals increase the exchanges dramatically.

3.2 The stage of weight assignment

This stage occurs in fuzzy logic controller. Fuzzy logic controller is the most important component of controlling. In this part, the weight of groups and nodes are determined and the decisions on load distribution are made.

In different algorithms, different criteria are considered for load assignment of the system. In fact, one of the problems with the classic algorithms is the lake of a proper criterion to measure load rate of the system. In different algorithms, criteria such as request arrival rate [19], average time for completing all the processing and the average response time for the nodes [7] or a combination of them has been considered as the measurement criterion. In this paper the system load and waiting time for processing the nodes in the time interval are used as input parameters of fuzzy controller. The waiting time of other nodes are measured in relation to maximum waiting time in a time interval.

Since fuzzy logic has been used and this logic is rule-based, seven fuzzy subsets have been considered for each input to determine the rules. By using these inputs, 77 rules are obtained, some of which will be discussed below:

1) If (current load is Vlow) and waiting time is Small) then (output1 is Nice)

2) If (current load is VHight) and (waiting time is Med) then (output1 is bad)

For instance, in rule1, if the current load is very low, and waiting time is small, fuzzy output which indicates the node state, would be excellent and its weight would be high and the number of sent tasks will be decided in higher proportion. In the rule 2, if the node is very heavily loaded and its waiting time is medium, the node state would be bad, and no more tasks will be send to it. In figure 3 membership function for the two inputs are shown.

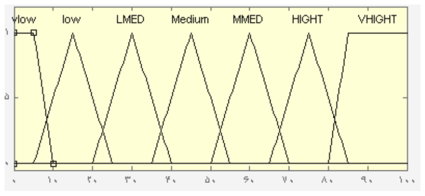

Fig3: membership function of fuzzy inputs

3.3 distribution stage

In this stage the distributer, distributes the requested processing load on the nodes based on their assigned weights. The global manager sends some of the requests to the distributors based on the assigned weights of groups. The distributors distribute the received tasks among the nodes, based on the weight of each node.

4. Simulation and comparing the efficiency of the proposed algorithm.

In this paper, the efficiency of the three algorithms of dynamic, random, and proposed fuzzy algorithm are compared, base on three criteria of drop rate, throughput, and average response time. To evaluate the efficiency of the algorithm, simulation is carried out in a variety of conditions. All simulation studies are done in OPNET simulation toolbox. The comparisons are made in two different conditions: i) based on different numbers of node with a constant number of tasks, and ii) for different numbers of tasks with a constant number of nodes. The memory and service rates for nodes of each group have been specified differently. General characteristics of the group are similar. In case 'ii' the number of nodes in each group is supposed 13. In case 'i' the number of tasks is supposed 44000 packets. Three task generators communicate with global manager through a communication link, delivering the request to it. Logical delays have been considered for communication links. The lengths of the packets are considered variable. To show the difference of algorithms, the task generators have been designed in a way that the entry rate the system would be unpredictable. This situation makes the difference between static and dynamic algorithms more distinct. The entry rate to the system is never constant in actual systems too. The following graphs show the resulting experiments in the OPNET software environment in different states.

Figure 4 shows the drop rate for 13 constant nodes with different tasks. As it can be seen, when there is a heavy load, fuzzy algorithm performs better than the other algorithms. Figure 5 denotes a state in which the number of the nodes varies between 15 to 39, and the number of task is 44000. As you may see, when the number of the nodes is small, all algorithms perform almost similar. But, as the number of the nodes increases, fuzzy algorithm, at a noticeable distance, has a less drop rate than that of dynamic and random algorithms. In general, in two state fuzzy algorithms has the less drop rate than two other algorithms. Figure 6 and 7 show throughput in constant node and variable task, and in variable node and constant task states respectively. In both conditions, fuzzy algorithm state performs better. Figure 8 shows the average response time. Clearly, fuzzy algorithm takes less time than the other algorithms. Finally, the results indicate that fuzzy algorithm has more efficiency and less response time compared to dynamic and random algorithms.

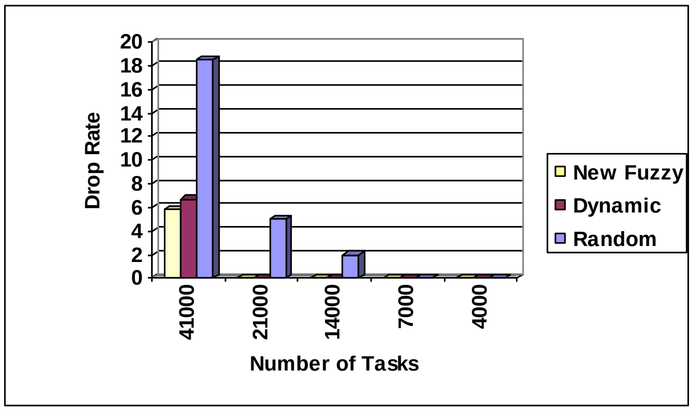

Fig 4: drop rate with different number of tasks and constant nodes

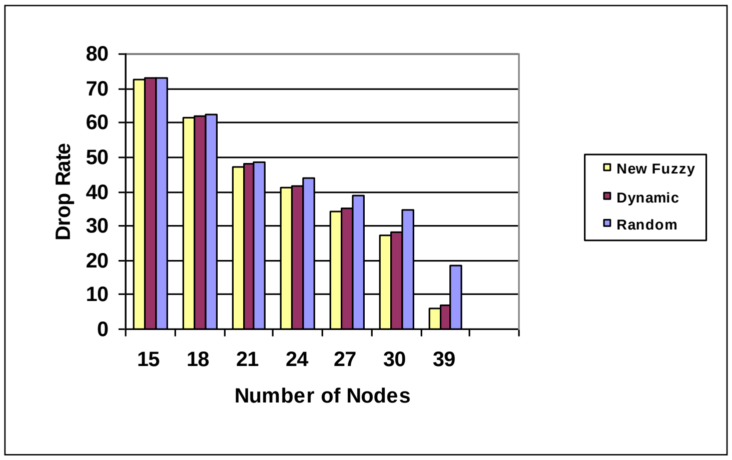

Fig 5: drop rate with constant number of tasks and different nodes

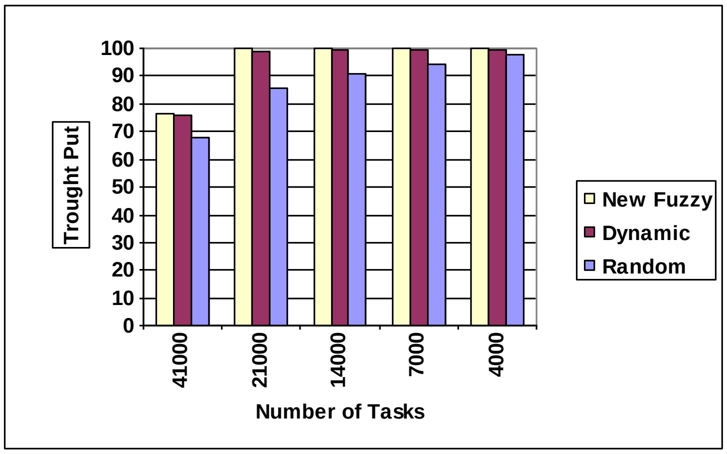

Fig 6: throughput with different number of tasks and constant nodes

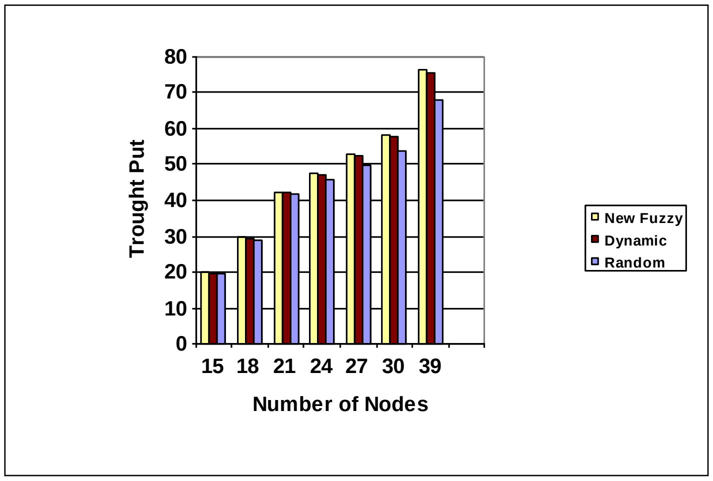

Fig 7: throughput with constant number of tasks and different nodes

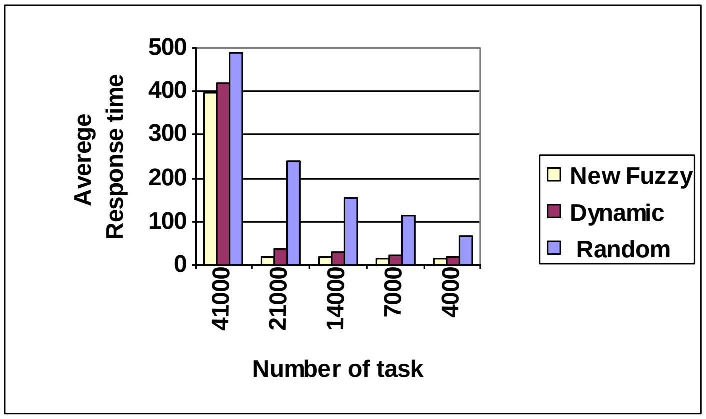

Fig 8: average response time

5. Conclusion

This paper provides a fuzzy logic-base intelligent load balancing in centralized distributed system. The efficiency of proposed load balancing is studied in OPNET simulation environment. The length of sent packet and service rate in each node has been considered non-constant. The current load of the system and waiting time for the last processed task in each node is considered as the fuzzy controller inputs. Based on the weight assigned for each node, a percentage of existing tasks is assigned to that node. The result of the experiments in the states of constant and variable nodes, and constant and variable tasks, indicates that this algorithm performs much better than both static and dynamic algorithms in terms of Throughput, Drop Rate and Response Time.

References

[1] Bonomi .F and Kumar .A,(1990), Adaptive Optimal Load-Balancing in a Heterogeneous Multiserver System with a Central Job Scheduler, *IEEE Trans. Computers*, vol.39(10), pp.1232-1250.

[2] Cheung L.-S, Kwok Y.-K, (2004), “On Load Balancing Approaches for Distributed Object Computing Systems”, *The Journal of Supercomputing*, pp. 149-175.

[3] CHEUNG L.SUN,(2000), A fuzzy approach to load balancing in a distributed object computing network, In Proc. *Of 6th Int IEEE Conf. HPDC*.

[4] Deneubourg J. L., Goss S., Franks N,(1990), The dynamics of collective sorting robot-like ants and ant-like robots. In From Animals to Animates, Proc. *of the First Int. Conference on Simulation of Adaptive Behavior*, pages 356-363. MIT Press.

[5] Gelenbe. E and Kushwaba .R (1994), Dynamic Load Balancing in Distributed Systems. *IEEE*.

[6] Huang .Ming-Chang,(2003),load balancing and congestion control using fuzzy logic control method in computer networks. *PHD Thesis*. The University of Wisconsin - Milwaukee.

[7] Joshi B. S., Hosseini Seyed H, (1993), a Methodology for Evaluating Load Balancing Algorithms, in Proc. *Of 6th Int. Conf. IEEE HPDC*.

[8] Kasabov, Nikola K,(1998), Foundations of neural networks, fuzzy systems, and knowledge engineering. *The MIT Press*.

[9] Krem O and Kramer J,(1992), Methodical Analysis of Adaptive Load Sharing Algorithms, *IEEE Trans. On High Performance Computing* Vol4 (3).

[10] Lin H.C and Raghavendra C.S,(1992), A Dynamic Load-Balancing Policy with a Central Job Dispatcher (LBC), *IEEE Trans. Software Eng*., vol.18 , no(2), pp.148-158.

[11] Lan Y and Yu T,(1995), A Dynamic Central Scheduler Load-Balancing Mechanism, Proc. *IEEE 14th Ann. Int'l Phoenix Conf. Computers and Comm*., pp.734-740.

[12] Ming K, Yu V, and Chou C,(2000), A Fuzzy-Based Dynamic Load-Balancing Algorithm, Proc. Of fuzzy System Application Symp. *IEEE Press*.

[13] Park Chulhye and kuhl Jon G,(1995), A Fuzzy-Based Distributed Load Balancing Algorithm for Large Distributed Systems. *Second International Symposium on Autonomous Decentralized Systems (ISADS'95)* p. 0266, IEEE.

[14] Schnekenburger Thomas,(2000), “Load Balancing in CORBA: A Survey of Concepts, Patterns, and Techniques”, *The Journal of Supercomputing*,pp.141-161

[15] Sandeep S. W,(2008), “Classification of dynamic load balancing strategies in a Network ofWorkstations”.*Fifth International Conference on Information Technology: New Generations.IEEE*. pp1005-1014.

[16] Veeravalli B. and Min W. Han,(2004), Scheduling Divisible Loads on Heterogeneous Linear Daisy Chain Networks with Arbitrary Processor Release Times, *IEEE TRANS. ON PARALLEL AND DISTRIBUTED SYSTEMS*, VOL.15(3).

[17] Zomaya A. Y and Teh Y, (2001), Observations on using genetic algorithms for dynamic loadbalancing, *IEEE Trans. on Parallel and Distributed Systems* 9 ,pp.899-911.

[18] Zomaya Albert Y. and Yee-Hwei,(2001), Observations on Using Genetic Algorithms for Dynamic Load-Balancing. I*EEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, VOL. 12, NO. 9.

[19] Zeng Z. and Veeravalli B,(2000), Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems, *10th Int. Conference on Parallel and Distributed Systems ,IEEE.*