A FUZZY-BASED DYNAMIC LOAD-BALANCING ALGORITHM

S. Zindhu
Assistant Professor, Department of Computer Science, Periyar University, St. Joseph's College of Arts and Science for Women, HOSUR – 635 126

ABSTRACT

Load Balancing means to distribute the load among the processors to improve the performance of a parallel and distributed system. This project presents a new fuzzy algorithm for dynamic load balancing in distributed system. This project presents a new fuzzy algorithm for dynamic load balancing in distributed environment. This approach characterizes the uncertainty in a distributed system by using the fuzzy set theory, fuzzy control methods and presents a fuzzy based distributed load balancing algorithm that explicitly reflects the effect of uncertainty in the decision making process. The notion of the linguistic variables is used to model the sender and receiver node. We have taken the two input variables, load and number of heavy load nodes. And one output variable status of load balance node. On the basis of decision rules we have concluded the status of load balance node. My work is to do propose an improved algorithm for dynamic load balancing in the distributed environment using fuzzy logic concept. The decision rules have implemented on the MATLAB. Then the system model is created on network simulation software. Simulation results show the response time.

KEYWORDS: Fuzzy Logic, Load Balancing, SLB, DLB, FLC, MATLAB.

1. Introduction

Distributed computer systems are becoming increasingly available because of the rapid decrease in hardware cost and the advances in computer networking technologies. It is frequently observed that in a computing environment with a number of hosts connected by a network, the hosts are often loaded differently. In typical distributed system task arrive at the different nodes in random fashion. It causes a situation of non-uniform load across the different nodes. It is observed by existence of nodes that are highly loaded while the others are lightly loaded or even idle. Some situation is harmful to the system performance in terms of response time and resource utilization. To overcome such problem by increasing the overall throughput of the system by allowing heavily loaded nodes to redistribute tasks to lightly loaded nodes. To achieve such a goal, system nodes must exchange state information and make control decisions.

A Load Balancing: Load balancing is a technique to spread work between two or more computers, network links, HDs, CPUs, or other resources, in order to get the optimal resource utilization, throughput or response time. Load Balancing is done by migrating tasks from the overloaded nodes to other lightly loaded nodes to improve the overall system performance. The processors are categorized according to workload in their CPU queues a heavily loaded (more tasks are waiting to be executed), lightly loaded (less tasks are waiting to be executed in CPU queue) and idle processors (having no pending work for execution). Here CPU queue length is used as an indicator of workload at a particular processor. The algorithms used for load balancing may require no information or only information about individual jobs or may make decisions based on the current load situation.

There are two types of load balancing algorithm:

Static Load Balancing (SLB): In this method the performance of the nodes is determined at the beginning of the execution. Then depending upon their performance the workload is distributed in the start by master node. The slave processors calculate their allocated work and submit the result to the master.

Dynamic Load Balancing (DLB): DLB algorithms that consider the current load conditions in making job transfer decisions. In this method the workload is distributed among the nodes at runtime. The master assigns a new process to the slave based on the new information collected. Dynamic load balancing does the process while job are in execution. The dynamic load at each host is calculated and redistributes the load on all the processors, when the load imbalance reaches some predefined level, this distribution of work take place. This redistribution does incur extra overhead at execution time.

Fig 1: Simple dynamic load balancing to avoid overload on heavily loaded node by transferring process to light weighted node.

Problem Definition: My work is to do propose an improved algorithm for dynamic load balancing in the distributed environment using fuzzy logic concept.

1. Concept Of Fuzzy Logic In Load Balancing

A fuzzy logic model, it performs dynamic load balancing is proposed. The model is based on fuzzy logic that takes imprecise information for its inputs. This information is passed to the fuzzy inference engine which performs the analysis and assigns different ranks to the services. Clients can then choose the most appropriate service based on the rank of service. The service attributes is used as the service rank which characterized the availability of services.

The fuzzy expert system reflects the impression in state information and makes scheduling decision based on a fuzzy logic. The distributed computing system cannot be improved beyond a limit which is determined by the degree of uncertainty in the decision making process. In this mechanism, each distributed node dynamically adjusts its thresholds denoting the amount of consistency relaxation depending on the degree of uncertainty in the system state. A fuzzy based consistency model provides a mechanism that allows each node to make flexible scheduling and state update decisions based on its threshold.

A. Fuzzy Logic

Fuzzy logic was developed by Zadeh. It represents a form of mathematical logic. Values between 0 and 1 represent uncertainty in decision-making. 0 indicates a false value. 1 indicates a true value. So, within a fuzzy set a value x is not restricted by the values 0 or 1, but from the real interval [0, 1].

Fuzzy logic is a logical system, which is the extension of multivalve logic. In a wider sense fuzzy logic is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of object with un-sharp boundaries in which mem-
b. Fuzzy Logic Controller
To tackle the load balancing problem, conventional control theory can be applied to restore system equilibrium. Fuzzy logic control attempts to capture intuition in the form of IF–THEN rules and conclusions are drawn from these rules.

Fig 2: Fuzzy Inference Engine
The fuzzifier is the input interface which maps a numeric input to a fuzzy set so that it can be matched with the premises of the fuzzy rules defined in the application specific rule base. The rule base contains a set of fuzzy if–then rules which define the actions of the controller in terms of linguistic terms. The fuzzy inference engine applies the inference mechanism to the set of rules in the fuzzy rule base to produce a fuzzy set output. This involves matching the input fuzzy set with the premises of the rules, activation of the rules to deduce the conclusion of each rule that is fired and combination of all activated conclusions using fuzzy set union to generate fuzzy set output to a crisp output. Based on the crisp output, the fuzzy logic controller can drive the system under control.

3. Distributed Load Balancing Models
In distributed model, every host has a local monitor associated. Each monitor collects and updates information about the status of the local host. The primary advantages of this model are high performance, availability and extensibility at low cost. Conventional algorithms of distributed load balancing including Random, Sender-Initiated, Receiver-Initiated and Symmetric Algorithm.

A. Random Algorithm
Random algorithm is the simplest one. In this algorithm each node checks the local workload during a fixed time period. When a node becomes over loaded after a time period, it sends the newly arrived job to a node randomly no matter the load of target node is heavy or not. Only the local information is used to make the decision. The random algorithm has the lowest overhead because of its simplicity and without negotiation with other hosts. It can relocalize the system load balancing very well.

B. Sender Algorithm
The sender algorithm is based on the sender policy. When a node becomes over-loaded after a period of time, it selects the target node randomly and looking for its load status which is under-loaded or not. If it is under-loaded, an ACCEPT message is feedback to original host, otherwise it replies a REJECT message. If the requesting node is still over-loaded when they ACCEPT reply arrives, the newly arrived task is transferred to the probed node, otherwise the task keeps executing locally. The mechanism seals to push a task from the requesting node to the probed node after a period of time checking.

C. Receiver Algorithm
The receiver algorithm is designed according to the receiver policy. Once if a host becomes under-loaded, the node will poll the information form any other node to check if it is over-loaded. When an overloaded nodes was found, an ACCEPT message is feedback, otherwise it replies a REJECT message. The migration of a task from the probed node is still under-loaded.

D. Symmetric Algorithm
In comparison with the sender algorithm and the receiver algorithm, the symmetric algorithm shows two-side effects. When a node becomes over-loaded, sender algorithm enabled, when it is under-loaded, the receiver algorithm is active. This algorithm is combination version of the sender and receiver algorithm. This model is adjusted based on the current load-level of the node by allowing the algorithm to switch automatically between sender and receiver algorithm. When the load status is over-loaded, it plays the role of the sender algorithm in contrast; it plays the role of the receiver algorithm.

4. Review Of Literature
1. As an area of research, load balancing has received considerable attention since the early days of distributed systems in 1970s.
2. A Fuzzy based distributed load balancing algorithm, Chulhye Park and Jon G.Kuhl, 1995 IEEE
3. A global plan policy for coherent co-operation in distributed dynamic load balancing algorithm, Mourad Kara, 1995

5. Implementation And Results
The Algorithm: We present the fuzzy based distributed load balancing algorithm.

The Algorithm basically divided into two parts: 1. The system Model II.

The Scheduler:
- a) Threshold Estimation
- b) Decision Making

The System Model
A distributed system is assumed to be a collection of autonomous nodes connected by a communication network. Message passing is the only form of communication among nodes. The system model with d dimensions has 2^d nodes. If we number the nodes from 0 thru (2^d)-1 and look at the numbers as d-digit binary numbers, then each node will be connected to other nodes which differ only in one bit. That is, suppose d=4 for example, then 0010 (node 2) will be connected to 1010 (node 10), 0110 (node 6), 0000 (0) and 0011 (node 3). This is how the hypercube is built in hubcened; the gate number is the number of the bit that differs, that is, node 0010’s gate #2 goes to node 0011.

The system model module in hubcened takes the d dimensional as a parameter. It also takes another parameter called node type: a string that names the module type to be used as node in the hypercube. In the net, when a hypercube network is described, nodetypes is give the “HypercubeNode” value. HypercubeNode is a compound module type which consists of a sender (source) and receiver (sink) and a router module.

The Scheduler has two functions, threshold estimation and decision making. When a scheduler is invoked, it estimates two numerical thresholds from the current states of uncertainty sources based on a fuzzy control base, and making scheduling and state update decision using fuzzy consistency model. Implementation of the Scheduler In fuzzy logic toolbox we have taken two input parameters. The first input parameter is ‘load’ and the second one is ‘Number of heavy load Node’ and one output i.e. ‘status of load balancing node’. We measure load and Number of heavy load node on a 0 to 10 scale and status of load balancing node on 0 to 20 scale.

We need to define fuzzy sets for the input parameters, ‘load’, and ‘number of heavy load node’ levels, and the output, ‘status of load balancing node’. For this we define five membership functions for first input parameter i.e. ‘load’ and two membership functions for second input parameter i.e. ‘number of heavy load node’. And two membership functions for output parameter ‘status of load balancing node.’

FIRST INPUT PARAMETER: Load (0-10)
M11: Very lightly
M12: lightly
M14: heavy
M15: very heavy

SECOND INPUT PARAMETER: No. of heavy load node (0-10)
M11: more
M12: less

OUTPUT PARAMETER: Status of load balance node (020)
M11: sender
M12: receiver

Define Decision Rules
Assuming sender initiated load balance algorithm proposed knowledge base is as follows:
Rule [1]. If (load is very_lightly_load) then (status_LOADBALANCE_node is receiver)

Rule [2]. If (load is very_heavy_load) then (status_LOADBALANCE_node is sender)

Rule [3]. If (load is heavy_load) and no.__heavy__load___nodes is less) then (status_LOADBALANCE_node is sender)

Rule [4]. If (load is heavy_load) and (no.__heavyload___nodes is more) then (status_LOADBALANCE_node is receiver)

Rule [5]. If (load is lightly_load) and (no.__heavy__load___nodes is more) then (status_LOADBALANCE_node is receiver)

Rule [6]. If (load is lightly_load) and (no.__heavy__load___nodes is less) then (status_LOADBALANCE_node is sender)

Rule [7]. If (load is moderate_load) and (no.__heavy__load___nodes is more) then (status_LOADBALANCE_node is receiver)

Rule [8]. If (load is moderate_load) and (no.__heavy__load___nodes is less) then (status_LOADBALANCE_node is sender).

REFERENCES


