A Genetic-Fuzzy Logic Based Load Balancing Algorithm in Heterogeneous Distributed Systems

Kun-Ming Yu *, Ching-Hsien Hsu and Chwani-Lii Sune
Department of Computer Science and Information Engineering
Chung-Hua University
Hsin-Chu, 300, Taiwan, R.O.C.
e-mail: yu@chu.edu.tw

Abstract
Distributed processing is recognized as a practical way to achieve high performance in various computational applications. Many dynamic load-balancing algorithms have been proposed for parallel and discrete simulations. But the actual performances of these algorithms have been far from ideal, especially in the heterogeneous environment. In this paper, a hybrid approach using fuzzy supervised learning and generic algorithm is presented. The fuzzy membership function is dynamically adjusted by the genetic coding. Moreover, the proposed load-balancing algorithm has learning capability. The experimental results show that our proposed algorithm has better performance comparing with other classical load balancing algorithms.

Key Words
Load-balancing, genetic algorithm, fuzzy logic, heterogeneous environment

1. Introduction
Load balancing in a distributed system is a process of sharing computational resources by transparently distributing system workload. With the advent of high-speed communication links, it has become beneficial to connect stand-alone computers in distributed manner through a high-speed link. The primary advantages of these systems are high performance, availability, and extensibility at low cost. Therefore, distributed computing has gained increasing importance in the recent as a preferred mode of computing over centralized computing. Many researches have proposed different kinds of approaches for the load balancing problem [10, 11, 15, 16].

A load balancing system is composed of three design issues: the information gathering policy, the negotiation policy and the migration policy [1]. Traditional strategies of the load balancing systems usually take advantage of some fix values to distinguish workload (e.g. over-loaded or under-loaded). Many load-balancing approaches based on this conjecture have been introduced in the past [1, 10, 11, 12, 15, 16, 17]. In conventional load balancing systems, resource indexes are necessary to be the input training data, and the output (threshold of workload) can be decided impersonally. But the output values are fixed; it cannot indicate the degree of the workload. Moreover, there exists a sharp distinction between members and non-members; the tasks reallocation action will be made frequently around the threshold. This will result in an unstable system and cause unnecessary overhead. Moreover, the workload estimation of each host is very difficult and time-consuming. To solve this problem, [16] proposed a fuzzy logical theory to estimate the load status of each node and apply fuzzy information rule to determine the number of tasks shall be migrated on a heavily load node.

The fuzzy logic offers a framework for representing imprecise, uncertain knowledge. Similar to the way, in which human beings make their decisions, fuzzy systems use a mode of approximate reasoning, which allows it to deal with vague and incomplete information. However, fuzzy systems have the problem of determining its parameters. Ones of the most important parameters of fuzzy system are the Membership Functions (MF). The fuzzy inference engine in conjunction with the control rules to determine an appropriate output response then uses the value ranges. In most fuzzy systems, the shape of MF of the antecedent, the consequent and fuzzy rules were determined and tuned through trial and error by human operators. Therefore it takes much iteration to determine and tune them. There are simple methods to turn MF such as Neural Networks [2], genetic algorithms (GA) were used as in [14], and the GA has give faster learning response than the neural networks.

Therefore, we purposed a genetic algorithm approach to
construct a fuzzy logic control distributed system. This fuzzy logic artificial intelligence setting adjusts controller parameters or membership functions by genetic algorithm, it will not only have the power to improve the efficiency of multitask migration but also have fuzzy parameter learning capability. The fuzzy membership function can be adjustable according to the change of system environment immediately.

The organization of this paper is as follows. Section 2 describes the related work including the load balancing approaches. We then discuss several famous load balancing algorithms and the fuzzy enhanced symmetric algorithm [17] in section 3. We then present the proposed scheme with genetic algorithm embedded fuzzy enhance symmetric algorithm in section 4. Section 5 states the implementation issue as well as the experimental results. The conclusion and future work are provided in the last section.

2. Related Work

Load balancing can be performed either statically or dynamically. Previous researches on static and dynamic load balancing can be found in [4, 7, 13], respectively. In static load balancing, the tasks are assigned to nodes by analyzing their past behaviors or only using some conventional rules, which are independent of the actual current system state. The principal advantage of static policies is its simplicity. There is no need to maintain and process the information about the system state. The results of the previous studies suggest that dynamic policies have greater potential for performance improvement than static policies. There was approach using fuzzy logic control to enhance the dynamic load-balancing algorithm had been published [17]. But the fuzzy membership function was defined by benchmark program offline. After the system running a long period of time, the system status may change a lot. Therefore, the fuzzy membership function of cannot reflect the real situation of the environment.

The workload collection is one of the most important issues in dynamic load balancing approach. The information collection policy denotes not only the amount of workload about the systems but also the information gathering rules used in making the tasks reallocation decision. The goal of this policy is to obtain sufficient information in order to make a decision weather the host’s load is heavy or not. We say that a good information gathering policy [6] should be able to predict the workload in the near future, relatively stable and have a simple (ideally linear) relationship with the resource indexes. But it is difficult in real world, especially in heterogeneous systems.

Many traditional load-balancing schemes used a threshold value as the information policy after load indices generated. If the load index is above (under) the threshold, the host is said to be over-loaded (under-loaded), we can find out that the load status is classified into only two states, heavily or lightly. This binary-state makes the system load state fluctuate between heavily or lightly load wildly when the workload is near the threshold value. It will cause the task reallocation frequently because of little load change. Some researchers added a tolerance range around the threshold value [15, 16].

3. The Dynamic Load Balancing System

The distributed systems can be characterized by distribution of both physical and logical features. The architecture of a distributed system is usually modular and consists of a possibly varying number of processing elements. An arbitrary number of system and user processes may be executed in the system. A process can usually be executed on various machines. There are a number of factors to be considered when selecting a machine for process execution. These factors may include resource availability and utilization of various resources. A dynamic load balancing policy can employ either centralized or distributed control.

3.1 Centralize Load Balancing Model

In this model [3], a processor was appointed to be the centre controller, which collects and updates the information about the state of every other processor in the system. When a node decides that a task is eligible for load balancing, it sends a request to the specified processor to determine the suitable placement of the task. The advantage of this architecture was task reallocation action could be done accurately. But in this scheme has a potential risk, if the centre controller crashed, the system cannot work anymore, and in a large system this information traffic can’t ensure deliver the host processor when the network is busy.

3.2 Distributed Load Balancing Model

In distributed model, every host has a local monitor associated. Each monitor collects and updates the information about the state of the local host. The primary advantages of this model are high performance, availability, and extensibility at low cost. Conventional algorithms of this model include Random, Sender-Initiated [5], Receiver-Initiated [5] and Symmetric Algorithm [1,17].

3.2.1 Random Algorithm

Among the algorithms, the Random Algorithm is the simplest one [1]. 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 heavily 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. However, it can’t reallocate the system load balancing very well.

3.2.2 Sender Algorithm

The Sender algorithm is based on the Sender policy [5]. 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 the ACCEPT reply arrives, the newly arrived task is transferred to the probed node; otherwise the task keeps executing locally. This mechanism seals to push a task from the requesting node to the probed node after a period of time checking.

3.2.3 Receiver Algorithm

The Receiver Algorithm is designed according to the Receiver policy [5]. 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.

3.2.4 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 [1]. In other words, 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 Receiver algorithm.

3.2.5 Fuzzy Enhanced Symmetric Algorithm

Most conventional information gathering policies use load indices with a threshold value to determine the load status of host. The major problem is how to define an appropriate threshold value. The fuzzy theory can improve the information policy would be more objective and flexibility [16] to be the migration policy has improved this shortcoming. Some researchers use fuzzy logic control to solve this problem [17]. In [17], the experimental results show that used the fuzzy inference rules to obtain the migrated task numbers.

The typical architecture of a Fuzzy Logic Control (FLC) is composed of four principal components: a Fuzzifier, a Fuzzy Rule Base, an Inference Engine, and a Defuzzier. The workload is determined by using FLC in [17]. According to the host’s status is over-loaded or under-loaded, negotiation policy would initiate to find the suitable host to make the task migration. If the target node was being found, the migration policy will take advantage of defuzzification to calculate to number of tasks to be migrated.

But this method still not good enough to build a good dynamic load-balancing environment, because of the parameters of FLC be inaccurate and cannot make the best decision of the migrated number of tasks. In order to overcome this shortage, therefore, we proposed a new scheme that embedded online genetic algorithm to tune the fuzzy membership function dynamically. It can adjust the membership function in terms of the feedback values dynamically and react the overall systems status immediately.

4. Genetic based fuzzy logic control system

The genetic algorithm (GA) is an optimization search algorithm. GA is known to be particularly suitable for learning in complex domains and hence can be used for structure and parameter adaptation in fuzzy system, but it takes a considerably long time to converge to a suitable solutions. The basic concepts of GA were developed by Holland [8, 9], and have subsequently been extended in several research studies. Typically the GA starts with little or no knowledge of the correct solution and depends entirely on responses from an interacting environment and its evolution operators to arrive at good solutions.

The GA processes imitate natural evolution, and hence include bio-mimetic operation such as reproduction, crossover, and mutation. A conventional GA has four features: population size, reproduction, crossover, and mutation. GA’s maintain a set of candidate solution called a population. Candidate solutions are usually represented as strings of fixed length, called chromosomes, coded with binary character set. The first step of GA is to generate an initial population by random in cycles called generations. The chromosome is represents by a binary string matrix depending upon the system condition. By applying the operators such as selection, crossover and mutation, the chromosome with the highest fitness is chosen to determine the population chromosome.

In the paper, the genetic algorithm was designed to adjust
the value of membership function of Fuzzy System. An increase in the number of input variables causes an exponential growth in the number of rules generated. We devise an online genetic algorithm (OGA for short) adaptive mechanism for updating of the associated parameters of fuzzy membership function dynamically.

4.1 Online Genetic Algorithm

In OGA processes such as crossover, reproduction and mutation will proceed in the usual manner. The following genetic operation operations are applied to each string:

(1) Coding: The coding of fuzzy membership functions in a chromosome is shown in Figure 1. A triangular membership function is used in the fuzzy set, a and b are representing the center and width of the membership function, respectively. The type of coding used in this research is the concatenated binary string by the position of membership function center and width.

![Figure 1: Coding of a fuzzy membership function](image)

(2) Population size: The choice of an appropriate population size is a fundamental decision to be taken in all GA implementations. If the population sizing is too small the GA will usually converge too quickly, and too large a population will take a very long time to evaluate. In the study, population sizes are set to 50 and each chromosome is 26 bits.

(3) Reproduction: Reproduction is the process through selecting two parent genes from the current population. Selection is based probabilistically on a gene’s fitness value; the higher the fitness of a gene, the more likely it can reproduce.

(4) Crossover: Crossover operates on two solution strings and results in another two strings. Typical crossover operator exchanges the segments of selected strings across a crossover point with probability. There are two steps produces two offspring by crossover operator. At first, two strings from the reproduced population are mated at random, and a crossover site is randomly selected. Then the strings are crossed and separated at the site. We used two point crossover-site for each parent strings with crossover probability Pc.

![Figure 2: Crossover operation](image)

(5) Mutation: The mutation operator prevents irreversible loss of certain patterns by introducing small random changes into chromosomes. Change each bit value with the probability Pm.

![Figure 3: Mutation operation](image)

(6) Fitness Function: The genetic algorithm is able to optimize the characteristics explicit in the fitness function. Here the fitness function using following formula:

\[ F = \frac{1}{\alpha \cdot RT^2 + \beta \cdot TT} \]

where 0 <= \( \alpha, \beta <= 1 \),

RT: the response time,

TT: the turnaround time.

The process of Online Genetic Algorithm for fuzzy control is presented in figure 4.

![Figure 4: Online Genetic algorithm for fuzzy control](image)
### 4.2 Structure of Genetic Based Fuzzy Logic Control Systems

There are four components in our proposed genetic based fuzzy logic control load balancing system: information module, negotiation module, migration module, and online genetic algorithm module. The information module defines the workload status of every host. Then the negotiation module will probe the target host to request the task reallocation action. The migration module will make the decision of the migrated number of tasks and move the tasks to the target host. Finally, the online genetic algorithm will adjust the center value of fuzzy membership function, if the workload was still heavy (or light). Moreover, the online genetic algorithm will evaluate the current fuzzy rules to meet the load index or not. The architecture of genetic based fuzzy logic control model is shown in figure 5.

![Architecture of Genetic Fuzzy Logic Control System](image)

Figure 5: The architecture of our proposed load balancing algorithm

### 5. Implementation and experimental results

In order to verify the performance of our proposed scheme we implement our algorithm in a distributed environment, called Java Load Balancing System [18], which is implemented by using Java language. The system supports heterogeneous, static load balancing and dynamic load balancing. We also implemented three other algorithms (random, receiver initial, and symmetric) for comparison. In order to verify that our proposed online genetic based fuzzy logic control load balancing algorithm will accomplish a high system performance. Six workstations running different operation systems. The operation systems including the Unix, Win NT and Win 98.

#### 5.1 Online Genetic Algorithm Parameter

The genetic operation should be used in a way that achieves high-fitness individuals in the population rapidly without leading to a total convergence. In the paper, we used partial-random method to achieve high-fitness for a short-time interval. In our experiment, the size of population is set to 50, the total generations is 1000, the probability of crossover =0.8, the probability of mutation is 0.02.

### 5.2 Experimental results

In the experiment, the task number is adjustable parameters. In the static load balancing part, we used Random dispatch model. Four different algorithms were implemented, including random, receiver initial, symmetric, and our proposed algorithm. We have not implemented the sender-initial algorithm, since its performance usually worse than receiver-initial policy. In the experiment, we compare the performance of average response time, average turnaround time and overall throughput.

#### 5.2.1 Response Time

The response time denotes the time from the submission of a task until the first response is produced. In figure 6, we can see that our algorithm can cut down the average response time at least 50% when the tasks number is 70.

![Average Response Time](image)

Figure 6: Response time

#### 5.2.2 Turnaround time

The second index can estimate the efficient of overall system is the turnaround time. The turnaround time is the interval from the time of a task submission to the time of completion. Figure 7 is the average turnaround time of four algorithms. In the figure, we can find out our algorithm has the smallest turnaround time.

#### 5.2.3 Throughput

The overall throughput under different load balancing schemes is also discussed in our experiment. The definition of throughput is the number of processes that are completed per time unit. In figure 8, we also can find that our proposed algorithm always keeps higher throughput when the tasks number is increased.
6. Conclusion
In this paper, we design and implement an intelligent dynamic load balancing algorithm based on online genetic based fuzzy logic control. In our research, the proposed algorithm can correctly evaluate the workload of each machine in the system and make a decision of the migrate task exactly. In the scheme, OGA can dynamically adjust the fuzzy membership function based on the feedback information.

The experimental results show that our proposed load balancing can indeed significantly reduce the response time and turnaround time as well as increasing overall throughput.

Reference