A Fuzzy-Based Dynamic Load-Balancing Algorithm

Kun-Ming V. Yu*, and Chih-Hsun Chou*

*Department of Computer Science and Information Engineering,

Chung-Hua University

Hsin-Chu, Taiwan 300

R.O.C.

Yao-Tien Wang†

†Department of Computer Science and Information Engineering

National Central University

Chungli, Taiwan 320

R.O.C.

ABSTRACT

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, we design and implement a load-balancing system based on fuzzy logic control. The fuzzy algorithm has been implemented in a loosely coupled distributed system. On-line of workload measure has been addressed herein as being the load information policy, negotiation policy, and migration policy. The experimental results indicate that the fuzzy-based load- balancing algorithm not only effectively reduces the amount of communication messages but also provides considerable improvement in overall performance such as short response times, high throughputs, and short turnaround times.

Key words: Fuzzy Logic Control, Dynamic Load Balancing, Distributed Computing System.

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 [1, 6, 7, 8, 9, 10, 11]. A distributed computing system comprises of software programs and data resources dispersed across independent computers and connected through a communication network. A workstation user may not use the machine all the time, but may require more than it can provide while actively working. Some hosts may be heavily loaded, while other remains idle. Performance enhancement is one of the most important issues in distributed systems. The performance of the system can often be improved to an acceptable level simply by redistributing the load among the hosts.

The load balancing schemes conceptually can be categorized into two types: *static* and *dynamic*. We say a scheme is static, if it applies load-balancing technique while dispatching tasks into computing hosts. However it does not satisfy the real case that the workload is fluctuant in the computing cluster. The dynamic load-balancing scheme reallocates the computing tasks according to the current workload of each host. Therefore, it has better performance comparing with static scheme. 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, 6, 7, 8, 9, 10, 11]. 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 resolve these problems, we propose a Fuzzy-based dynamic load balancing scheme for evaluating the workload of each host as well as determining a suitable destination host to receive (send) jobs. In the scheme, we adopt run-queue length and CPU utilization as the input variables for fuzzy sets and define a set of membership function. It has been shown that our proposed load balancing algorithm not only effectively reduced the amount of communication messages but also provides considerable improvement in overall performance such as short response times, high throughput, and short turnaround times.

The remainder of this paper is organized as follows. Section 2 describes the proposed fuzzy-based load balancing algorithm in detail and the system structure, respectively. Section 3 presents the experimental results. Finally, the conclusion is given in Section 4.

2. Distributed Load Balancing Models

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 distributed load balancing including *Random* [1], S*ender-Initiated* [1], *Receiver-Initiated* [1] and *Symmetric Algorithm* [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.

2.2 Sender Algorithm

The Sender algorithm is based on the Sender policy [3]. 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.

2.3 Receiver Algorithm

The Receiver Algorithm is designed according to the Receiver policy [3]. 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.

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. 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. Fuzzy Logic Control-based Load Balancing System

The design of Fuzzy Logic Control consists of four modules: a fuzzy rule base, a fuzzy inference engine, fuzzification, and defuzzification. The linguistic approach of system modeling can be formulated in three distinguishing features: the use of linguistic variables in place of or in addition to numerical variables; the characterization of simple relations between variables by IF-THEN fuzzy rules; the formulation of complex relations by fuzzy reasoning algorithm.

The structure of a load balancing system is composed of three design phases: the information policy, the negotiation policy and the migration policy [7]. By applying fuzzy logic control to these three design phases, we can effectively raise the overall performance in a distributed system. We can construct different run queue length membership function, CPU utilization membership function, and center value for linguistic labels around through fuzzy c-means clustering algorithm according to various platform characteristics capacity.

The migration policy pertains to managing the migration of tasks form one host to another. The numbers of migrated tasks are according to the value calculated by MAX-MIN composition from the CPU utilization and queue length. Measurements of input variables of a fuzzy controller must be properly combined with relevant fuzzy information rules. The purpose of defuzzification is to convert each result obtained from the inference engine, which is expressed in terms of fuzzy sets, to a single real number. We used centroid method because it supports software real time fuzzy controls to distinct the difference of workload on two machines.

3.1 Information Policy

The information policy indicates the significance of various information regarding the system. From which, information gathering fuzzy rules is used to determine the system workload is heavy or not. Many researchers use CPU queue length as the single load index for a host in distributed system. Although CPU queue length is the obvious factors impacting on the system load, there are other factors also influencing the computing load, such as CPU utilization, disk I/O, memory paging, network, etc. For the accuracy of evaluating the load status of a host, we employ the CPU queue length and CPU utilization as the input variables for fuzzy sets. Fuzzification function is introduced for each input variable to express the associated measurement uncertainty. The information gathering process can be started periodically or in dependence on significant changes of system states.

3.1.1 Membership Function

A fuzzy set can be defined mathematically by assigning to each possible individual in the universe of discourse a value representing its grade of membership in the fuzzy set. These membership grades are represented by real-number values ranging between 0 and 1 and fuzzy implication, T, is a function of form $T:[0,1] \times [0,1] \rightarrow [0,1]$. The support of a fuzzy set A contains all element of x that has a nonzero membership grade. $Support(A) = \{x \in X \mid u_A(x) > 0\}$. We assume that the range of CPU queue length and CPU utilization fuzzy sets represent linguistic concepts system load as Light, Medium, and Heavy. Fuzzification function is introduced for each input variable to express the associated measurement uncertainty. Figure 1 and 2 shows the 3-level linguistic concepts system load of CPU queue length membership function and CPU utilization membership function respectively.

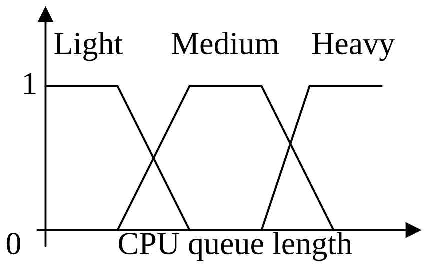

Figure 1: CPU queue length membership function

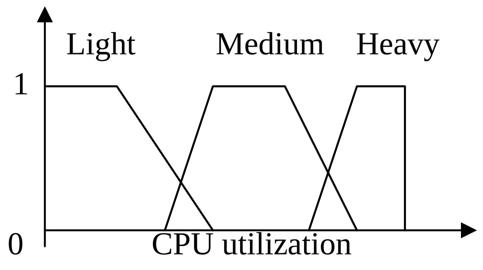

Figure 2: CPU utilization membership function

3.1.2 Inference engine

Measurements of input variables of a fuzzy controller must be properly combined with relevant fuzzy information rules to make inferences regarding the system Load State. In our system with variables CPU queue length, CPU utilization, we may proceed as follows. The purpose of the fuzzification function is to interpret measurements of input variables, as an example, a fuzzification fCPU queue length applied to variable CPU queue length. Then, the fuzzification function has the form fCPU queue length:$[0,n] \rightarrow R$, where R denotes the set of all fuzzy numbers, and fCPU queue length (X_0) is a fuzzy number chosen by fCPU queue length as a fuzzy approximation of the measurement CPU queue length = X_0. It is obvious that, if desirable, other shapes of membership function may be used to represent the fuzzy numbers fCPU queue length (X_0). For each measurement CPU queue length = X_0, the fuzzy fCPU queue length(X_0) enters into the fuzzy inference process. We convert the given fuzzy inference rules of the form CPU queue length membership and CPU utilization membership function into equivalent simple fuzzy conditional propositions of the form:

Rule 1: IF (CPU queue length, CPU utilization) is A_1 x B_1, then system load is C_1
OR
Rule 2: IF (CPU queue length, CPU utilization) is A_2 x B_2, then system load is C_2
OR
…………………………………………………………………….
OR
Rule n: IF (CPU queue length, CPU utilization) is A_n x B_n, then system load is C_n
Fact: (CPU queue length, CPU utilization) is fcpu queue length (x_0) x fcpu utilization (y_0)

Conclusion: System load is C.

3.2 Negotiation Policy

The negotiation policy selects the host to or form which tasks will be migrated when the load reallocation event takes place. The knowledge pertaining to the given control problem is formulated in terms of a set of fuzzy inference rule. We use three load actions, which are send-state (heavy), stabilize-state (moderate), and receive-state (light).

3.3 Migration Policy

The migration policy pertains to handle the migration of tasks form one host to another. The numbers of migrated tasks are according to the value calculated by MAX-MIN composition from the CPU utilization and run queue length. Measurements of input variables of a fuzzy controller must be properly combined with relevant fuzzy information rules. The purpose of defuzzification is to convert each result obtained from the inference engine, which is expressed in terms of fuzzy sets, to a single real number. We used centroid method because it supports software real time fuzzy controls to distinct the difference of workload on two machines. This value is calculated by the formula:

$$Y^0 = \sum_{i=1}^{n} W_i * B_i \;/\; \sum_{i=1}^{n} W_i. \qquad (1)$$

Where $\mathbf{Y^0}$ = output of fuzzy control,
W_i = the antecedent degree of ith control rule and
B_i = the consequent center value of ith control rule

Figure 3 depicts the architecture of the Fuzzy-based dynamic load-balancing algorithm.The typical architecture of a FLC, which is composed of four principal components: a *Fuzzifier*, a *Fuzzy Rule Base*, an *Inference Engine*, and a *Defuzzifer*. In the information policy, the *fuzzifier* has the effect of transforming crisp measured data into suitable linguistic values. Then the *fuzzy rule base* stores the empirical knowledge of the operation of the process of the domain experts. Moreover, the *inference engine* is the kernel of FLC, and it has the capability of simulating human decision making by performing approximated reasoning to achieve a desired control strategy. After these steps, the workload would be decided. According to the host is over-loaded or under-loaded, negotiation policy would initiate to find the suitable host to make the task migration. Finally, if the target node was found, the migration policy will take advantage of defuzzification to calculate the number of migrated tasks.

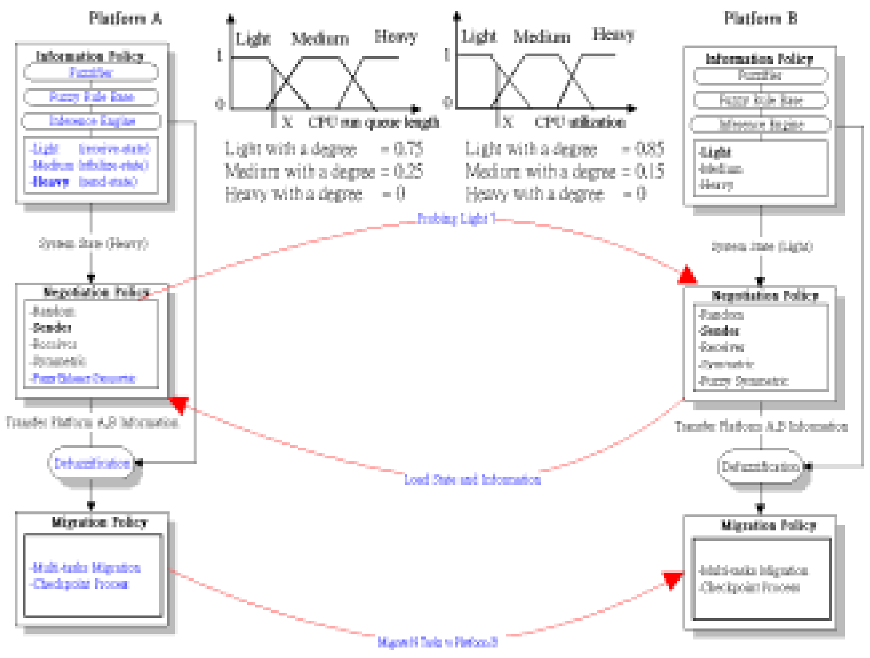

Figure 3: The overall structure of Fuzzy-based load balancing algorithm.

4. Experimental Results

In order to show that our FLC-based load balancing algorithm will accomplish a high system performance, we use Java programming language to construct a distributed load balancing computing environment. Since Java language is suitable to support heterogeneous distributed computing system. We use six workstations running different UNIX Operating System to build a heterogeneous computing environment. These workstations are located on different NFS and connected by communication network.

The experimental results reveal that the proposed load balancing system yields better performance as compared with others conventional algorithm. Figure 4 shows the response time of different load balancing schemes in the cases of different work sizes. We found that our proposed load balancing scheme has the shortest response. Figure 5 presents the turnaround time of different load balancing schemes in the cases of different work sizes. From the figure, we found that our proposed load balancing scheme has the shortest turnaround time. Especially, our proposed scheme performs well when the number of jobs is large. The throughput is also discussed in our experiments. Figure 6 shows that our proposed scheme has the highest throughput. For the reducing of unnecessary communication message, Figure 7 indicates that our algorithm has the smallest number of request message among all load balancing approaches.

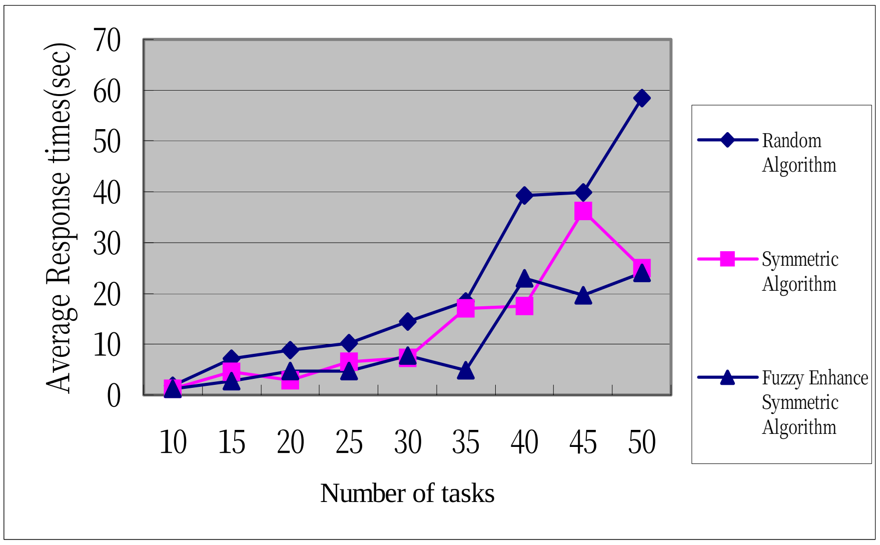

Figure 4: The response time of different load balancing schemes

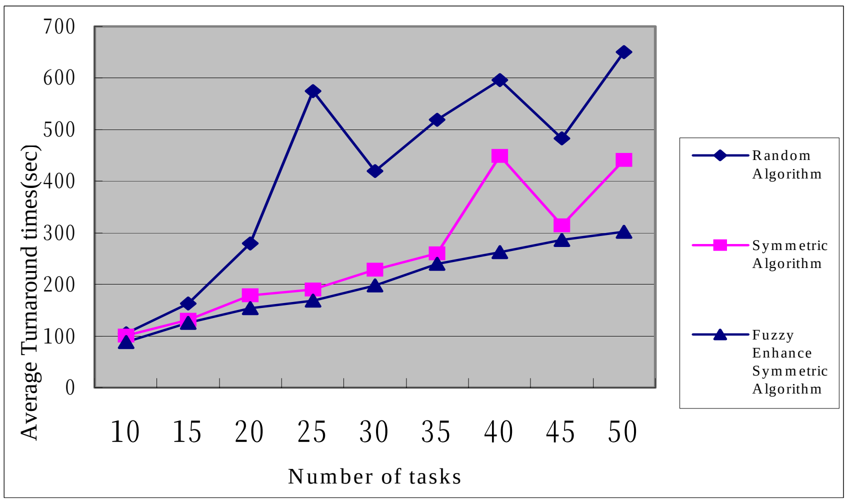

Figure 5: The turnaround time of different load balancing schemes

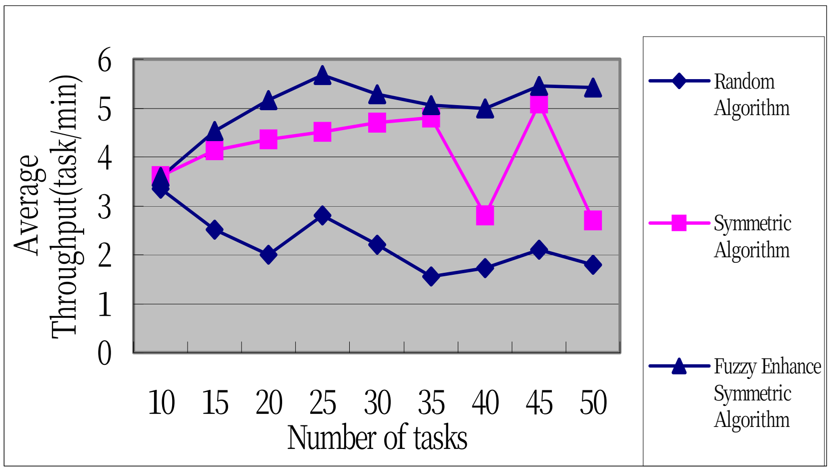

Figure 6: The Throughput of different load balancing schemes

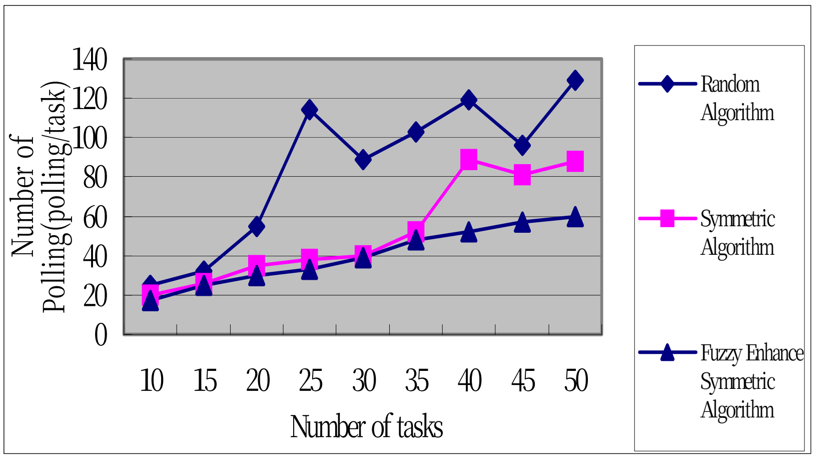

Figure 7: The number of messages sent of different load balancing schemes

4. Conclusion

We have proposed a new dynamic load balancing scheme based on Fuzzy Logic Control. The FLC-based algorithm is used to correctly evaluate the load status of a host in heterogeneous computing system. It also can efficiently determine the suitable host for migrating jobs. The performance of proposed scheme is better than that of the conventional schemes on the turnaround time and throughput. It also can effectively eliminate the unnecessary communication messages between hosts in distributed system.

Reference

[1] K. Benmohammed-Mahieddine, P. M. Dew, and M. Krar, " A periodic Symmetrically-Initiated Load Balancing Algorithm for Distributed Systems", *IEEE ICPDS*, 1994, pp. 616-621.

[2] Y-C Chow and Walter H. Kohler, "Model for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," IEEE Trans. on Computers, vol. C-34, No. 3, March 1985, pp. 204-217 .

[3] D. L. Eager, E.D. Lazowska, and J. Zahorjan, " A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing," *Proc. Of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Aug. 1985, pp. 1-3.

[4] Mohammad R.Emami, I.burhan Turksen, and Andrew A. Goldenberg, "Development of A Systematic Methodology of Fuzzy Logic Modeling," IEEE Transactions on Fuzzy System, vol 6, no 3, Aug. 1998, pp. 346-360.

[5] D. Ferrari and S. Zhou, An Empirical Investigation of Load Indices for Load Balancing Applications, Tech. Rep.UCB/CSD/87/353, Computer Science Division, Univ. of California, Berkeley, CA, 1987

[6] H-C Lin and C. S. Raghavendra, "A Dynamic Load Balancing Policy with a Central Job Dispatcher," IEEE Trans. On Parallel and Distributed System, Jul. 1991, pp. 264-271.

[7] Margaret Schaar, Kemal Efe, Lois Delcambre, and Laxmi N. Bhuyan, "Load Balancing with Network Cooperation", IEEE Trans. On Parallel and Distributed System, Jul. 1991, pp. 328-335.

[8] K-M Yu, Siman J-W. Wu, and Tzung-Pei Hong. " A Load Balancing Algorithm Using Prediction", IEEE Computer Society. The Second Aizu International Symposium on Parallel Algorithms/Architecture Synthesis. March 17-21, 1997, pp. 159-165.

[9] K-M Yu, Siman J-W. Wu, and Tzung-Pei Hong. "An Adaptive Load Balancing Algorithm in Homogeneous System", Proceedings of 1997 Workshop on Distributed System Technologies & Applications, pp. 496-503.

[10] K-M Yu and L-K Wang. “A Dynamic Load Balancing Algorithm Using Artificial Neural Network”, Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, July 27-31, 1997, pp. 364-367.

[11] Kun-Ming Yu, Yau-Tien Wang and Chih-Hsun Chou, “A Dynamic load balancing Approach using Fuzzy Logic Control”, Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing (ACS’99), August 9-12,1999.