International Journal of Computer Applications (0975 – 8887)
Volume 101– No.10, September 2014

A Genetic-Fuzzy Algorithm for Load Balancing in Multiprocessor Systems

Roya Nourzadeh
Department of Computer Engineering, Germi branch, Islamic Azad University, Germi, Iran

Mehdi Effatparvar
ECE Department, Ardebil Branch, Islamic Azad University, Ardebil, Iran

ABSTRACT

With the increasing use of computers in research contributions, added requirement for faster processing is now an important necessity. Parallel Processing describes the concept of running tasks which can be run simultaneously on several processors. Load balancing is very important problem in multiprocessor systems. In this paper, we introduce a approah based on Genetic Algorithms and Fuzzy Logic for laod balancing in parallel multiprocessor systems that call GAF algorithm. Extensive simulation shows our algorithm is better than other approach. Simualation results indicate our algorithm have maximum utilization and it reduce total response time of system.

General Terms

Scheduling, Genetic Algorithms, Fuzzy Logic.

Keywords

Load Balancing, Multiprocessor systems, Genetic Algorithm, Fuzzy Logic, Idle Time, Load Balancing Ratio..

1. INTRODUCTION

In multiple processing, multiple processors come together to implement a program. The major application of the systems is for problem solving in modeling and engineering sciences. Today, not just scientific problems solving requires parallel processing, but in addition some commercial applications require fast computers. number of these applications require the processing of large volumes of complex information. Some of these programs include data mining operations, medical imaging, oil exploration and etc[1-4].

Operating systems manage processors and other computer resources without the visible effect to the end user[5]. The multiprocessor capability increases the complexity of Operating systems and raises large amount of challenges to Operating systems such as for example simultaneous execution of tasks, scheduling, synchronization, processor failure, memory management and etc[6,7]. The majority of the modern Operating systems, like Unix, Linux, Microsoft Windows and Mac Operating system can handle running on multi-processor computers[8,9].

The scheduler of the Operating system has an essential role in the machine performance and processors utilization. For instance in the multiprocessor systems, each processor could have own queue for tasks, scheduler selects of this tasks to run. This may cause an imbalance one of the processors and the load balancing algorithm must certanly be defined to stabilize the load among the processors[9]. A periodic load balancing is commonly used to test the existence of imbalance among the processors. However, the periodic load balancing might not prevent the imbalance among the processors because of the the low response time of the load decrease. Therefore, this paper proposes a genetic-fuzzy algorithm for load balancing to prevent unnecessary idle periods, to stabilize the load among the processors, and to increase performance.

In this paper, we introduce a method based on genetic algorithm and fuzzy logic for scheduling tasks on multiprocessor systems. The main objective is to achieve performance and power improvement. Our method minimize the response time and idle time, also it maximum the utilization. Results of the simulations indicate minimum response time and idle time in comparison with other algorithms. This study is divided into the following sections: In section 2 an overview of Fuzzy Logic. Section 3 presents System setup in detailed. The proposed method is described in Section 3.2. Results of the study are analyzed in Section 4. Finally, Section 5 presents the conclusions.

2. REVIEW OF PREVIOUS WORKS

Several load balancing proposals are presented for the multiprocessor systems. Merkel and et al. proposes an optimal scheduling policy in multiprocessor system with Linux 2.6 kernel. The goal of the proposed algorithm is to determine the energy characteristics of tasks and to prevent overheating of an individual task, i.e., tasks are transferred from the overheated processor to the cooler processor. Overheating may reduce processor's clock frequency. However, the addition of task migrations break processor affinity and cause some performance penalties even if they may be negligible compared to the performance boost[10].

Correa and et al. [11] introduce a multi-level load balancing algorithm in NUMA systems for running in Linux operating system. Proposed algorithm detect the number of memory access levels in a device and builds an n-level scheduling domain hierarchy rather than Linux's two level memory access. This kind of implementation does not take full benefit of the proper machine's topology but the average performance improvement was as high as 10%.

Caprita and et al. [12] presents Grouped Distributed Queues. The goal of the paper is to reach accurate proportional fairness in scheduling with O (1) scheduling overhead. Grouped Distributed Queues works on the simple exponential grouping strategy and two levels hierarchical scheduler to prepare processes into groups. Grouping is dependant on simple processor time allocation rights and aggregate group shares. Based on simulations and kernel measurements, the presented algorithm provides good proportional fairness and scales relatively well with a big number of processors and processes. Zomaya and et al. [13]

introduces dynamic and centralized load balancing algorithm based on genetic algorithm. The key objectives are minimum response time, maximum processor utilization, and a optimal balanced load across all processors. The proposed algorithm use of decimal encoding mechanism to initialize a population and to make a string. Furthermore, the algorithm works on the fitness function for the performance measurements and the roulette wheel approach for a selection operator. Starting point of the crossover operation is selected randomly, and the swap mutation is applied. The proposed algorithm is very efficient, especially in the case of a big number of processes.

3. SYSTEM SETTING

In this section we describe system model in this paper. Model of system for Multiprocessor is as following:

- There are N_{n+1} number of processors and are fully connected.
- Communication and computation can be executed simultaneously. This can be accomplished by following the non-blocking send and non-blocking receive protocol for communication purpose.
- Before scheduling a particular task, the system assumed to know the runtime of that task.
- Processors are homogeneous.
- Tasks are independent.

It is assume that the communication between the processors will take place with the help of message passing interface environment.

4. GENETIC-FUZZY ALGORITHM FOR LOAD BALANCING

In this section, we introduce a method based on genetic algorithm and fuzzy logic for tasks scheduling on multiprocessor systems. The main objective is to achieve maximum utilization and load balancing among processors or resources. In fact, we use of Genetic Algorithm (GA) for scheduling, fitness function of GA is base on Fuzzy Logic. In the following we explain the our approach.

The objective of this study is to discover a sequence of tasks for minimizing total execution time and maximum utilization in parallel multiprocessor systems. Thus, each chromosome is sequence variety of tasks. Each task is considered as a gene. Therefore, the best way to encode chromosomes is permutations encoding. To start, GA should generate an initial random population for entry into the first generation. For this, a random generator function of chromosomes must be employed. Random chromosomes generate the initial population.

The important part of GA is the fitness function. The fitness function is defined over the genetic representation, and measures quality of the chromosomes. The fitness function is always dependent on the problem. In this paper, the fitness function is base on Fuzzy Logic. For fitness function, introduces the fuzzy approach that input variables are the sum load of ready queues. The dynamic ranges of the input variables (or membership functions) were divided into three part, i.e., low, medium and high, see Figure 1.

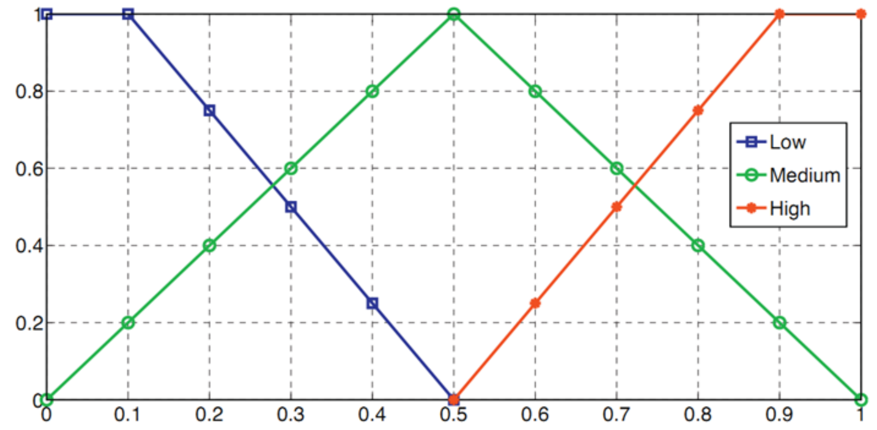


Figure 1: Membership functions of the linguistic variable sum load of ready queues.

The label low in membership functions defines that the sum load of all ready queues is so minor, so the load balancing is not needed. The label medium defines a range in which there is so much load.

Therefore, all processors should be running without idle time and utilization of processors are in maximum level. The membership function for the label high defines a range in which there is hug of work-loads in the system. The rules for load balancing were defined as fuzzy conditional statements, for example If load is low or medium then load balancing is not required. If load is high load balancing is required.

We consider for each prossecor tow value, if load of processor is lightly (low) or heavy (high) then p(i) = 0. Also if load of processor is moderate (normal) then p(i) = 1. So the fitness function is calculated according to the (1) equation:

$$F = \sum_{i=1}^{n} P(i) \tag{1}$$

P(i) is Processor i and n is complete number of processors. The fitness values have been evaluated for all chromosomes and the probability of higher fitness is to be selected for reproduction from current generation to the next generation.

For selection operator, we used tournament operator for selection. Two of chromosomes in population select randomaly. Between two chromosomes, chromosome which has a higher fitness is selected. This operation is repeated a number of population size.

For crossover operator, we used one point crossover for crossover. Two chromosome with crossover probability select for crossover. For mutation, we use bit fliping strategy that change gens of chromosome with mutation probability. Any gene with 1/c change. Means c is all number of genes (chromosome length). Also, we used generational strategy for replacing new population

With previous population. In fact, the new population replacing with all previous population in generational strategy. Genetic algortithm repeat until achieve to acceptable solution or the number of generations considered.

5. EVALUATION AND SIMULATION RESULTS

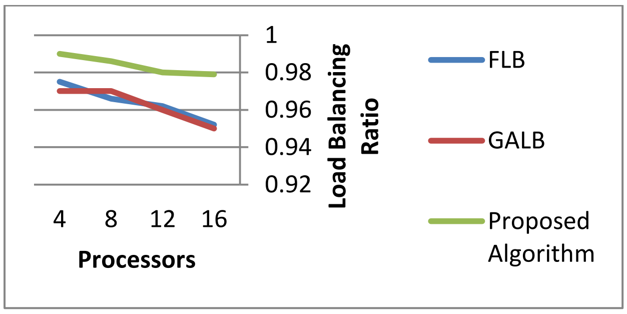
In this section, the results of the simulation that are employed to evaluate our algorithm are presented. Proposed algorithm is compared with two different load balancing algorithms, namely GALB and FLB. GALB proposed in [5] and it is only based on Genetic algorithm. FLB proposed in [9] and it is only based on Fuzzy Logic. We have used the MATLAB simulator to evaluate the proposed algorithm. Several experiments are considered to evaluate the algorithms. Experimental results show that the proposed algorithm performs better than GALB and FLB in all of the scenarios. We considered two scenarios. This scenarios shows in table 1.

Table 1: Scenarios of simulations

Scenario	Number of Tasks	Number Processors	Task	Processors
1	80	8	Independent	Homogeneous
2	500	16	Independent	Homogeneous

5.1 Evaluation of Load Balancing Ratio

Figure 2 and Figure 3 shows load balancing ratio of algorithms in both scenarios. According to figure 2 and figure 3, load balancing ratio of proposed algorithm is better than GALB and FLB in both scenarios.



Figur 2: Load Balancing Ratio in scenario 1.

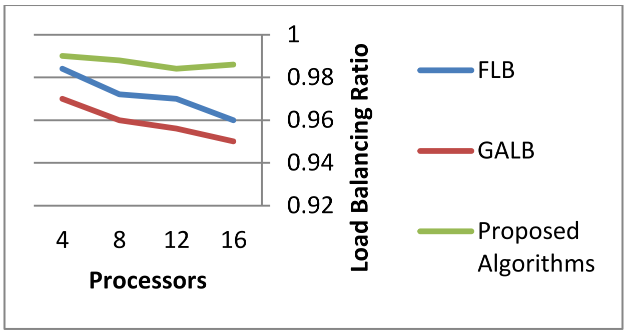


Figure 3: Load Balancing Ratio in scenario 2.

5.2 Evaluation Idle Time of Processors

In this section, we evaluate idle time of algorithms in all scenarios. Table 2 shows average total idle time of system in both scenario, respectively. According to obtained results in table 2, proposed algorithm is better than GALB and FLB in both scenarios. Because, idle time of processors in proposed algorithm are lesser.

Table 2: Average total idle time of system in both scenario

	Proposed Algorithm	**GALB**	**FLB**
Fraction of average idle time in scenario 1	%2	%6	%6
Fraction of average idle time in scenario 2	%4	%8	%11

5.3 Evaluation of System Utilization

Table 3 and table 4 presents performance results of system utilization for load balancing algorithms in scenario 1 and 2. According to obtained results, system utilization of proposed algorithm in both scenario is better. The proposed algorithm can have maximum utilization comoare to GALB and FLB.

Table 2: Average of system utilization in scenario 1

	Proposed Algorithm	**GALB**	**FLB**
System Utilization	0.99	0.98	0.96

Table 3: Average of system utilization in scenario 2

	Proposed Algorithm	**GALB**	**FLB**
System Utilization	0.98	0.96	0.95

6. CONCLUSIONS

The aim of the paper was to present the developed on demand based load balancing algorithm using Genetic Algorithm and Fuzzy Logic for the homogeneous multiprocessor environment. According to the simulation results, proposed algorithm has a better performance compared to other algorithms in all of the scenarios. We evaluate our algorithm with GALB and FLB algorithm. The performance metrics we use are the load balancing ratio, idle time and system utilization. Proposed algortihm in all performance metrics was better than GALB and FLB.

As future work, (1) we want investigate the performance of our algorithm under other scenarios. For example, when processors are heterogeneous, number of processors or number of tasks are very much and etc. (2) we want investigate the performance of our algorithm in Grid or Cloud systems.

7. REFERENCES

[1] Riky Subrata, Albert Y. Zomaya, Bjorn Landfeldt. Artificial life techniques for load balancing in computational grids. Journal of Computer and System Sciences 73 (2007) 1176–1190.

[2] Pratyay Kuilab, Prasanta K. Janaa. Energy Efficient Load-Balanced Clustering Algorithm for Wireless Sensor Networks. 2nd International Conference on Communication, Computing & Security. (2012) 771 – 777.

[3] Timur Keskinturk , Mehmet B. Yildirim, Mehmet Barut. An ant colony optimization algorithm for load balancing in parallel machine swith sequenc e-dependent setup times. Computers & Operations Research 39 (2012) 1225 –1235.

[4] Satish enmatsaa, Anthony Chronopoulos. Game-theoreticstaticloadbalancingfordistributedsystems. J.ParallelDistrib.Comput. 71(2011) 537–555.

[5] Hui Cheng, Shengxiang Yang, Jiannong Cao. Dynamic genetic algorithms for the dynamic load balanced clustering problem in mobile ad hoc networks. Expert Systems with Applications 68 (2012) 132–144.

[6] Primoz Rus, Boris tok, Nikola Mole. Parallel computing with load balancing on heterogeneous distributed systems. Advances in Engineering Software 34 (2003) 185–201.

[7] A. Saffar , R. Hooshmand , A. Khodabakhshian. A new fuzzy optimal reconfiguration of distribution systems for loss reduction and load balancing using ant colony search-based algorithm. Applied Soft Computing 11 (2011) 4021–4028.

[8] Maha A. Metawei, Salma A. Ghoneim ,Sahar M. Haggag. Load balancing in distributed multiagent computing systems. Ain Shams Engineering Journal (2012) 3, 237–249.

[9] Mika Rantonen, Tapio Frantti, Kauko Leiviska. Fuzzy expert system for load balancing in symmetric multiprocessor systems. Expert Systems with Applications 37 (2010) 8711–8720.

[10] Merkel, A., & Bellosa, F. (2006). Balancing power consumption in multiprocessor systems. SIGOPS Operating Systems Review, 40(4), 403–414. ISSN 0163-5980.

[11] Corrêa, M., Zorzo, A., & Scheer, R. (2006). Operating system multilevel load balancing. In SAC ’06: Proceedings of the 2006 ACM symposium on applied computing . 1-59593-108-2 (pp. 1467–1471).

[12] Caprita, B., Nieh, J., & Stein, C. (2006). Grouped distributed queues: Distributed queue, proportional share multiprocessor scheduling. In PODC ’06: Proceedings of the 25th annual ACM symposium on principles of distributed computing .1-59593-384-0 (pp. 72–81).