International Journal of Scientific Research Engineering & Technology (IJSRET)
Volume 1 Issue 5 pp 289-293 August 2012 www.ijsret.org ISSN 2278 - 0882

A Fuzzy Approach Scheduling on More Than One Processor System in Real Time Environment

[1]Mr. Sheo Das, [2]Ms. Payal Goel, [3]Ms. Kawaljeet Kaur
[1, 3] Research Scholar, Shrinathji Institute for Technical Education, Meerut (U.P.)
[2]Research Scholar, DVSIET, Meerut
Sheo.das@gmail.com, payalsmile86@gmail.com, kawaljeet29@rediff.com

ABSTRACT
Many scheduling algorithms have been studied to guarantee the time constraints of real-time processes. Scheduling decision of these algorithms is usually based on parameters which are assumed to be crisp. However, in many circumstances the values of these parameters are vague. The vagueness of parameters suggests that we make use of fuzzy logic to decide in what order the requests should be executed to better utilize the system and as a result reduce the chance of a request being missed. Our main contribution is proposing a fuzzy approach to multiprocessor real-time scheduling in which the scheduling parameters are treated as fuzzy variables.

In real-time systems, tasks have to be performed correctly and timely. Finding minimal schedule in multiprocessor systems with real-time constraints is shown to be NP-hard. Although some optimal algorithms have been employed in uni-processor systems, they fail when they are applied in multiprocessor systems.

The intrinsic uncertainty in dynamic real-time systems increases the difficulties of scheduling problem. To alleviate these difficulties, we have proposed a fuzzy scheduling approach to arrange real-time periodic and non-periodic tasks in multiprocessor systems. Static and dynamic optimal scheduling algorithms fail with non-critical overload. In contrast, our approach balances task loads of the processors successfully while consider starvation prevention and fairness which cause higher priority tasks have higher running probability.

Keywords*: Fuzzy logic, real-time scheduling, multiprocessor system.*

I. INTRODUCTION
Many applications namely avionics, traffic control, automated factory, and military systems require real time communication and computation. In real-time systems, all tasks have specific parameters such as deadline, priority, etc. Modern embedded computing systems are becoming increasingly complex [1]. Many real-time systems are hard and missing deadline is catastrophic [2-5]. A schedule which is executing all real-time tasks within their deadlines and all the other constraints are met, is called a feasible schedule [6]. Real-time scheduling can be classified in two categories, static [7] and dynamic [8] scheduling. Nonetheless, scheduling is more significant in real-time systems than non-real-time systems [9, 10, 15-20]. In real-time systems, tasks have to be performed correctly and in a timely fashion as well [11]. Tasks are classified as periodic and non-periodic [12, 13]. The execution requests of a periodic task repeatedly occur at regular intervals. On the contrary, execution requests of a non-periodic task are unpredictable.

Nowadays, using of real-time multiprocessor systems is dramatically increasing. Unfortunately, less is known about how to schedule multiprocessor-based real-time systems than that for uni-processors [14]. Having more computational complexity in practical algorithm cause wide range of algorithm's response time hence, deterministic timing behavior is the most important parameter for system's robustness especially in hard real-time system[21-24]. This behavior cause decrease in utilization of the system when unpredictable conditions happened.

The performance of a scheduling algorithm is measured in terms of additional processor required to be added at a schedule without deadline violations as compared to optimal algorithm [25]. In [26] it has been proved that finding a minimal schedule for a set of real-time tasks in multiprocessor system is NP-hard. Multiprocessor scheduling techniques in real-time systems fall into two general categories: partitioning and global scheduling [27]. Under partitioning, each processor schedule tasks independently from a local ready queue. Each task is assigned to a particular processor and is only scheduled

on that processor. In contrast, all ready tasks are stored in a single queue under global scheduling.

In the global scheme, processors repeatedly execute the highest priority tasks available for execution. It has been shown that using this approach with common priority assignment schemes such as rate monotonic (RM) and earliest deadline first (EDF) can give rise to arbitrarily low processor utilization [28]. In this approach a task can migrate from one processor to another during execution.

II. SCHEDULING ALGORITHMS

CPU scheduling is nothing but selecting a single process from a bunch of processes from ready queue. This assignment is carried out by software known as a scheduler. In order to measure the efficiency of a scheduling algorithm there are some criteria's such as Throughput (No. of processes completed per unit time), Turnaround time (The interval from the time of submission of a process to the time of completion), waiting time (The time spend by the process in the ready queue), Response time (The time interval between the submission of job until the first response is produced). Various traditional algorithms that we used earlier are.

A. FCFS:

First come first serve, is the simplest scheduling algorithm, FCFS simply arrange the processes in the order that they arrive in the ready queue. FCFS is a non preemptive scheduling algorithm. So there are fewer contexts switching overhead occurs. Throughput can be low, since long processes can hold the cpu. Turnaround time, waiting time and response time can be high for the same reason. No prioritization occurs, thus this system has trouble to meet deadlines of the processes.

B. SJF:

Shortest Job First is a scheduling policy that selects the waiting process with the smallest execution time to execute next. It's the most efficient process that we ever meet. However, it has a drawback of process starvation for process which will require a long time to complete if short processes are keep arrived continuously. It uses past behavior to indicate which process to run next, based on an estimate of its execution time. Shortest job next scheduling is rarely used outside of specialized environments because it requires accurate estimations of the runtime of all processes that are waiting to execute.

C. Priority scheduling:

The operating system assigns a fixed priority rank to every process, and the scheduler arranges the processes in the ready queue in order of their priority. Lower priority processes get interrupted by incoming higher priority processes. Overhead is not minimal, nor is it significant in this case. Waiting time and response time depend on the priority of the process. Higher priority processes have smaller waiting and response times. Deadlines can be easily met by giving higher priority to the earlier deadline processes. Starvation of lower priority processes is possible if large no of higher priority processes keep arrived continuously.

D. HRRN:

Highest Response Ratio Next **s**cheduling algorithm proposed by Brinch Hansen is to avoid limitations of SJF algorithm. It is similar to Shortest Job Next (SJN) in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting in the ready queue. Job which gain higher priority the longer they wait, which prevents process starvation. In fact, the jobs that have spent a long time in waiting can compete against those jobs estimated to have short run time.

$$\text{Response Ratio} = \frac{Waiting\ \ Time + Run\ \ time}{Run\ \ Time}$$

III. FUZZY INFERENCE SYSTEMS

Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth which denotes the extent to which a proposition is true. Whereas classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces Boolean truth values with a degree of truth. Degree of truth is often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision.

Fuzzy Inference Systems (FIS) are conceptually very simple. They consist of an input, a processing, and an output stage. The input stage maps the inputs, such as frequency of reference, recency of reference, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a corresponding result. It then combines the results. Finally, the output stage converts the combined result back into a specific output value [6].

An inference engine tries to process the given inputs and produce an output by consulting an existing knowledgebase.

There are two common inference processes [6]. First is called Mamdani's fuzzy inference method proposed in 1975 by Ebrahim Mamdani [8] and the other is Takagi-Sugeno-Kang, or simply Sugeno, method of fuzzy inference introduced in 1985 [9]. These two methods are the same in many respects, such as the procedure of fuzzifying the inputs and fuzzy operators. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant but Mamdani's inference expects the output membership functions to be fuzzy sets.

IV. THE PROPOSED MODEL

The block diagram of our inference system is presented in Figure 1.

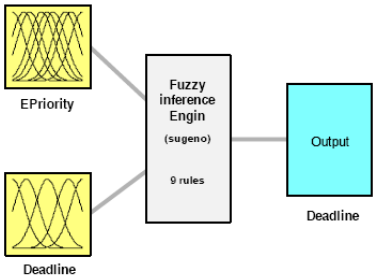


Figure1: Inference system block diagram

In the proposed model, the input stage consists of two linguistic variables. The first one is an external priority which is the priority assigned to the task from the outside world. This priority is static. One possible value can be the tasks interval, as rate monotonic algorithm does. For Figure 1, the other input variable is the Deadline. This input can easily be replaced by laxity, wait time, or so on, for other scheduling algorithms. Each parameter may cause the system to react in a different way. The only thing that should be considered is that by changing the input variables the corresponding membership functions may be changed accordingly. For the simulation purposes, as it is discussed later, four situations are recognized: First, by using laxity as a secondary parameter and, second, by replacing the laxity parameter with deadline and both of them are considered in partitioned and also global scheme. In fact, four algorithms are suggested: FGEDF1, FGMLF2, FPEDF3, and FPMLF4.

The output of the system is priority that determines which is used as a parameter for making a decision. Fuzzy rules try to combine these parameters as they are connected in real worlds. Some of these rules are mentioned here:

- If (EPriority is high) and (laxity is critical) then (Priority is very high)
- If (EPriority is normal) and (laxity is critical) then (Priority is high)
- If (EPriority is very low) and (laxity is critical) then (Priority is normal)
- If (EPriority is high) and (laxity is sufficient) then (Priority is normal)

In fuzzy inference systems, the number of rules has a direct effect on its time complexity. So, having fewer rules may result in a better system performance.

The Proposed Algorithm

The FGEDF algorithm is as follows:
FGMLF is much the same with FGEDF just by replacing the word deadline by laxity.
FPMLF is much the same with FPEDF just by replacing the word deadline by laxity.

Algorithm FGEDF

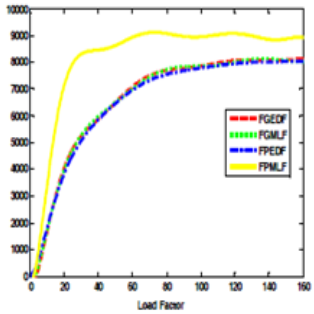
Loop // System is running for ever
For each CPU in the system do the followings:
1. for each ready task T (a task which is not running), feed its external priority and deadline into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)
End
End loop

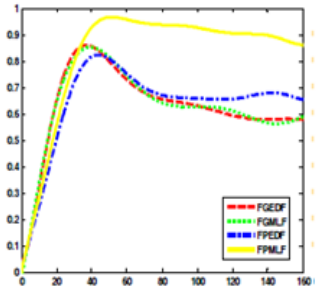
Algorithm FPEDF for each CPU

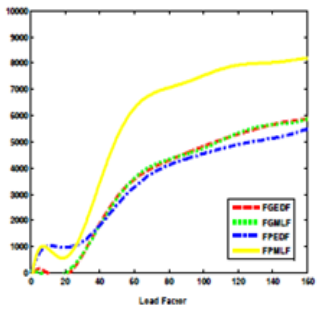
Loop
1. For each ready task T (a task which have not been run on another CPU), feed its external priority and deadline into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)
End loop

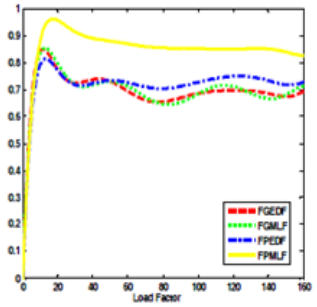
V. PERFORMANCE EVALUATION

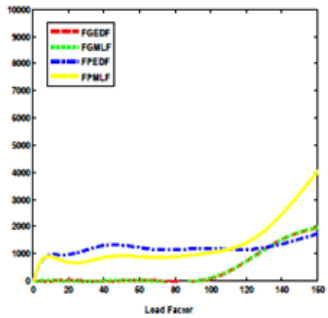
To evaluate our algorithm and to demonstrate its strength, 1024 test cases with different load factors were generated. In each test case, the number of tasks and the corresponding execution time and request interval were randomly generated. The behavior of all the four algorithms is compared with each other. Performance metrics, which are used to compare different algorithms, must be carefully chosen to reflect the real characteristics of a system.











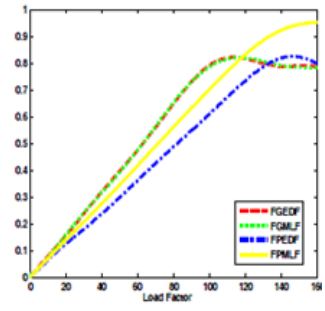


Figure 2: Number of misses for 7, 33,127 CPUs respectively
Figure 2. Number of misses for 7, 33,127 CPUs respectively

Figure 3: Average CPU utilization for 7, 33,127 CPUs respectively

These metrics are as follows. Number of missed deadlines is an influential metric in scheduling algorithms for soft real-time systems. Yet another metric, which is considered in our study, is the average CPU utilization. The main goal of a scheduling algorithm is to assign and manage system resources so that a good utilization is achieved. Among the four algorithms FGEDF and FGMLF nearly achieve the same performance in all situations and all metrics. FPMLF performs poorly in number of misses, but its performance on CPU utilization is much better than the others especially when the number of CPUs increase.

Now, we will compare the algorithms in each metric. Comparing number of misses, as Figure 2 shows, FPMLF has larger amount of misses and FPEDF seems to have fewest numbers of misses the other two algorithms carry out nearly the same. These figures show that deadline is a better parameter in this case. FPMLF which was the worse algorithm in first metric has the best CPU utilization among the other algorithms. This algorithm utilizes almost 100 hundred percent of CPU. As Figure 3 states, partition approach achieves better CPU utilization.

VI. CONCLUSION AND FUTURE WORKS

This paper has described the use of fuzzy logic to multiprocessor real-time scheduling. As it was shown, using deadline as a fuzzy parameter in multiprocessor real-time scheduling is more promising than laxity. The proposed scheduler which proposed in this paper has low complexity due to the simplicity of fuzzy inference engine. This model is efficient when system has heterogeneous tasks with different constraints.

In the future, we will conduct a deeper simulation and compare the results of fuzzy approach with the other algorithms.

REFERENCES

[1] F. Gruian, "Energy-centric scheduling for real-time systems," in Department of Computer Science. Ph.D dissertation: Lund University, 2002.

[2] S. M. Petters, "Bounding the execution time of real-time tasks on modern processors," in Proc. 7th Intl. Conf. Real-Time Computing Systems and Applications, Cheju Island, 2000.

[3] W. Lifeng and Y. Haibin, "Research on a soft real-time scheduling algorithm based on hybrid adaptive control architecture," in Proc. American Control Conf, Lisbon, Portugal, 2003.

[4] P. A. Laplante, "The certainty of uncertainty in real-time systems," IEEE Instrum. Meas. Mag., vol. 7, Dec 2004.

[5] J. Kreuzinger, A. Schulz, M. Pfeffer, T. Ungerer, U. Brinkschulte, and C. Krakowski, "Real-time scheduling on multithreaded processors," in Proc. 7th Intl. Conf.

Real-Time Computing Systems and Applications, Cheju Island, South Korea, 2000.
[6] N. D. Thai, "Real-time scheduling in distributed systems," in Proc. Intl. Conf. Parallel Computing in Electrical Engineering, Warsaw, Poland, 2002.
[7] C. Lin and S. A. Brandt, "Efficient soft real-time processing in an integrated system," in Proc. 25th IEEE Real-Time Systems Symp., 2004.
[8] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," IEEE Trans. Comput., vol. 51, Mar 2002.
[9] R. Jiminez-Peris, M. Patino-Martinez, and S. Arevalo, "Deterministic scheduling for transactional multithreaded replicas," in Proc. 19th IEEE Symp. Reliable Distributed Systems, Nurnberg, Germany, 2000.
[10] S. Zhiao, E. Jeannot, and J. J. Dongarra, "Robust task scheduling in non-deterministic heterogeneous computing systems," in Proc. IEEE Intl. Conf. Cluster Computing, Barcelona, Spain, 2006.
[11] M. Sabeghi, M. Naghibzadeh, and T. Taghavi, "Scheduling nonpreemptive periodic tasks in soft real-time systems using fuzzy inference," in Proc. 9th IEEE Intl. Symp. Object and Component- Oriented Real-Time Distributed Computing, Gyeongju, KOREA, 2006.
[12] G. Chen, G. Chen, O. Ozturk, and M. Kandemir, "An adaptive locality-conscious process scheduler for embedded systems," in Proc. 11th IEEE Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, 2005.
[13] S. A. Brandt, S. Banachowski, L. Caixue, and T. Bisson, "Dynamic integrated scheduling of hard real-time, soft real-time, and non-realtime processes," in Proc. 24th IEEE Intl. Real-Time Systems Symposium, Cancun, Mexico, 2003.
[14] G. Ascia and V. Catania, "A general purpose processor oriented to fuzzy reasoning," in Proc. 10th IEEE International Conf. Fuzzy Systems, Melbourne, Australia, 2001.
[15] J. Goossens, P. Richard, Overview of real-time scheduling problems. Euro Workshop on Project Management and Scheduling, 2004.
[16] M. Sabeghi, M.Naghibzadeh, T.Taghavi. A Fuzzy Algorithm for Scheduling Soft Periodic Tasks in Preemptive Real Time Systems. International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE), 2005.
[17] M.Sabeghi, M.Naghibzadeh, T.Taghavi, Scheduling Non-Preemptive Periodic Tasks in Soft Real- Time Systems Using Fuzzy Inference. 9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing (ISORC),2006.
[18] J. Shata Kadhim, M. Kasim Al-Aubidy : ComputerEng. Dept, Al- Blaqa''Design and Evaluation of a Fuzzy Based CPU schedulilnlg Algorithm'' Applied University, Al-Salt, Jordan Computer Eng. Dept, Philadelphia University, Amman, Jordan,Springer-verlag Berlin Heidelberg 2010.
[19] W. Stallings : Operating Systems Internals and Design Principles, 5th edn. Prentice-Hall,Englewood Cliffs 2004.
[20] C. Yaashuwanth, R. Ramesh: ,"Design of Real Time Scheduler Simulator and Devlopment of Modified Round Robin Architecture ",IJCSNS ,VOL.10 No.3,March ,2010.
[21]C. Lin and S. A. Brandt, "Efficient soft real-time processing in an integrated system," in Proc. 25th IEEE Real-Time Systems Symp,2004.
[22] B. Shahzad, , M. T. Afzal, "Optimized Solution to Shortest Job First by Eliminating the Starvation". In: The 6th Jordanian Inr. Electrical an Jordan , 2006.
[23] A J. Trivedi and Dr. P. S. Sajja ,"Improving efficiency of round robin scheduling using neuro fuzzy approach " ,IJRRCS vol.2,No. 2,April 2011
[24] M. Hamzeh,S. M. Fakhraie and C. Lucas ,"Soft real time fuzzy task scheduling for multiprocessor systems",world academy of science,engineering and technology vol.28, 2007.
[25] A. Silberschatz, J. L Peterson, and B. Galvin, Operating System Concepts, Addison Wesley, 7th Edition, 2006.
[26] S. A. Tanenbaum and S. A. Woodfhull , Operating Systems Design and Implementation, Second Edition, 2005
[27] W. Stallings, Operating Systems Internal and Design Principles, 5th Edition , 2006.
[28] D. Simon, Training fuzzy systems with the extended Kalman filter, Fuzzy Sets and Systems, vol. 132, No. 2, 1, December 2002.