Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.348 – 355

RESEARCH ARTICLE

REAL-TIME SCHEDULING OF SOFT PERIODIC TASKS ON MULTIPROCESSOR SYSTEMS: A FUZZY MODEL

Reshu Manwani, Nisha Phogat
CSE & MDU, India
CSE & MDU, India
manwani81@gmail.com, n.phogat@gmail.com

Abstract- Many scheduling algorithms have been studied to assure 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. A simulation is also performed and the results are judged against each other. It is concluded that the proposed fuzzy approach is very promising and it has the potential to be considered for future research.

Keywords- Fuzzy logic, multiprocessor real-time scheduling, FGEDF, FGMLF, FPEDF, FPMLF

I. INTRODUCTION

Scheduling real time systems involves allocation of resources and CPU-time to tasks in such a way that certain performance requirements are met. In real-time systems scheduling plays a more critical role than non-

real-time systems because in these systems having the right answer too late is as bad as not having it at all [1]. Such a system must react to the requests within a fixed amount of time which is called deadline.

In general, real-time systems can be categorized into two important groups: hard real-time systems and soft real -time systems. In hard real-time systems, meeting all deadlines is obligatory, while in soft real-time systems missing some deadlines is tolerable.

In both cases, when a new task arrives, the scheduler is to schedule it in such a way that guaranties the deadline to be met. Scheduling involves allocation of resources and time to tasks in such a way that certain performance requirements are met.

These tasks can be classified as periodic or a periodic. A periodic task is a kind of task that occurs at regular intervals, and a periodic task occurs unpredictably. The length of the time interval between the arrivals of two consecutive requests in a periodic task is called period.

Schedulability of periodic, preemptive, soft real-time tasks on uniprocessor systems is well understood; simple and efficient algorithms are available and widely used [2, 3, 4, 5].

Nevertheless, for multiple processors these algorithms are not promising. Meeting the deadlines of real-time tasks in a multiprocessor system requires a scheduling algorithm that determines, for each task in the system, on which processor it must be executed, and in which order with respect to the other tasks, it must start its execution.

Multiprocessor scheduling techniques in real-time systems fall into two general categories: partitioning and global scheduling [11]. 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. A single system-wide priority space is assumed: the highest priority task is selected to execute whenever the scheduler is invoked. Partitioning is the favored approach because it has been proved to be efficient and reasonably effective when popular uniprocessor scheduling algorithms such as EDF and RM are used [7]. But finding an optimal assignment of tasks to processors is NP-hard.

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 [7]. In this approach a task can migrate from one processor to another during execution.

In both cases researchers made some significant contributions by those results in better multiprocessor scheduling algorithms.

Although, these algorithms focus on timing constraints but there are other implicit constraints in the environment, such as uncertainty and lack of complete knowledge about the environment, dynamicity in the world, bounded validity time of information and other resource constraints. In real world situations, it would often be more realistic to find viable compromises between these parameters. For many problems, it makes sense to partially satisfy objectives. The satisfaction degree can then be used as a parameter for making a decision. One especially straightforward method to achieve this is the modeling of these parameters through fuzzy logic. The same approach is also applied on uniprocessor real-time scheduling in [12, 13].

The scope of the paper is confined to scheduling of soft periodic tasks in multiprocessors real-time systems. In section 2 the fuzzy inference systems are discussed. Section 3 covers the proposed model and section 4 contains the experimental results. Conclusion and future works are debated in Sections 5.

II. Fuzzy Inference Engine

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].

As discussed earlier, the processing stage which is called inference engine is based on a collection of logic rules in the form of IF-THEN statements where the IF part is called the "antecedent" and the THEN part is called the "consequent". Typical fuzzy inference systems have dozens of rules. These rules are stored in a knowledgebase. An example of a fuzzy IF-THEN rule is: IF *laxity* is *critical* then *priority* is *very high*, which laxity and priority are linguistics variables and critical and very high are linguistics terms. Each linguistic term corresponds to membership function.

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.

III. THE PROPOSED MODEL

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

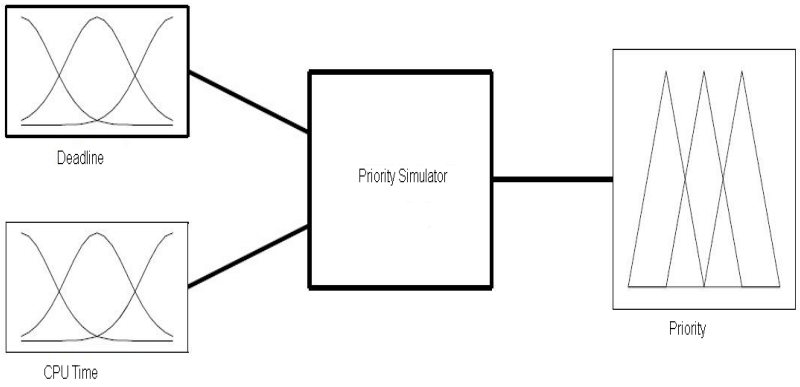

Figure 1. Inference system block diagram.

In the proposed model, the input stage consists of two linguistic variables. The first one is deadline which is the maximum time limit in which the task has to get executed. The CPU time is the time required by the task to get executed on assigned processor. Another possible input values can be the tasks interval, as rate monotonic algorithm does. For Figure 1, the other input variable is the Priority. 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.

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 (CPU Time is high) and (deadline is critical) then (Priority is very high)
- If (CPU Time is normal) and (deadline is critical) then (Priority is high)
- If (CPU Time is very low) and (deadline is critical) then (Priority is high)
- If (CPU Time is high) and (deadline is average) 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.

In fuzzy inference systems, the number of rules has a direct effect on its time complexity. Therefore, having fewer rules may result in a better system performance. However, depending upon the problem, rules may be increase to have optimal solution.

The proposed Mamdani Inference Engine is a Black box representation of simulation. In the process the researcher has the freedom to decide upon the input variables, membership functions and even define the rule set to define the working of the system, but the actual implementation follows data abstraction from the side of the inference engine.

When multiple tasks are to get executed, the basic concept of processor switching and time slices comes into picture.

In our proposed algorithm as shown under, a newly arrived task will be added to the input queue. This queue consists of the remaining tasks from last cycle that has not yet been assigned

Loop

For each processor in the distributed system, do the following:

1. **For each ready task, feed its CPU time, and deadline into the inference engine. Consider the output of inference module as priority of the task T.**
2. **Store the values of Priority in an array P().**

3. **Execute the task with highest priority until a scheduling event occurs. (a running task finishes, a new task arrives)**
4. **Update the system states.**

End Loop

After the successful completion of the algorithm, we shall get the task with maximum value of 'priority' and the same will be executed first followed by the next highest value of 'priority'. The cycle repeats itself till all scheduled tasks get executed by the available processors.

IV. EXPERIMENTAL RESULTS

Once all the tasks get executed by the available processors, researcher can generate the decision surface to see the value of the output variable (i.e. 'priority') under the combinations of different input parameters (i.e. 'CPU time', and 'Deadline'). The decision surface thus obtained is shown in the Fig. 2.

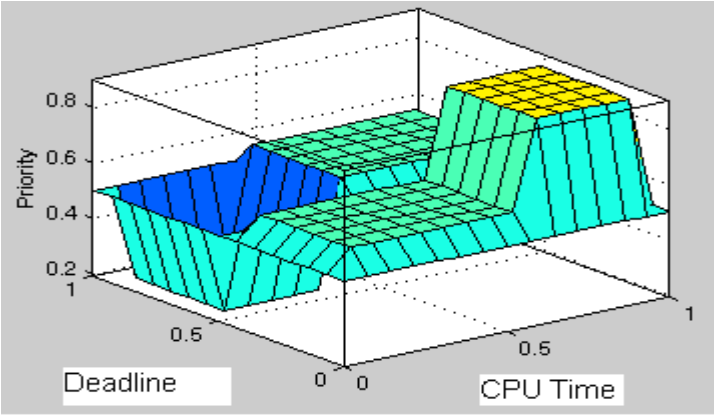

Fig. 2 Decision Surface

Choosing number of rules and membership functions directly affects system accuracy while performance of the system increases with rule size decrease. There are some techniques for adjusting membership functions however; in this paper we did not consider these approaches. Simulation results show that by increasing the number of system's processors, generation of high priority tasks increases until high priority task's waiting times is reduced to an acceptable range (Fig. 3). By increasing processor, high priority tasks have higher probability of execution while their deadline would not decrease to critical region. This behavior results in more execution for low priority tasks in medium load cycles. Next, low priority tasks generation increases to

handle low priority task's waiting time. Simulation results show that the model can feasibly schedule tasks when system load increases and keep system processors loads close to one even at crowded times.

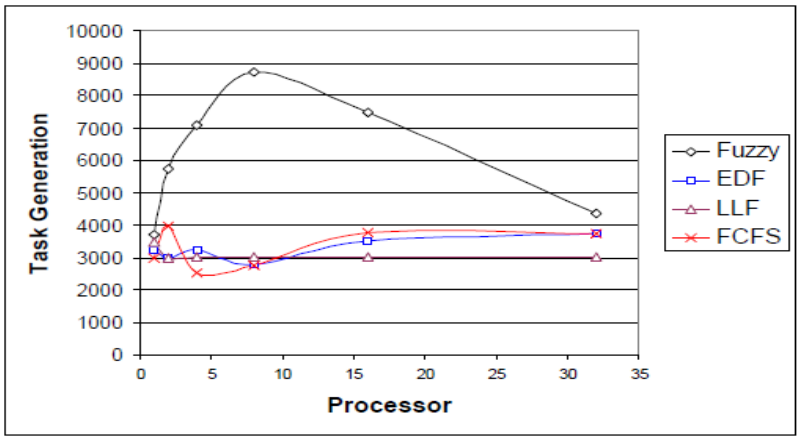

Fig. 3 Comparison with Traditional Approaches

Finding a minimal schedule for a set of real-time tasks in distributed systems is NP-hard [14]-[14]. However, other algorithms like LLF and EDF break down when the system is overloaded. In this model, process is independent of the number of system's processors. We note that processor's load remain always below one because of dispatcher's processing time. By analysis of task scheduler, periodic task's period increases automatically by scheduler with consideration of their priority and CPU time.

REFERENCES

[1] Ramamritham K., Stankovic J. A., 1994. Scheduling algorithms and operating systems support for real-time systems. *Proceedings of the IEEE*, Vol. 82(1), pp55-67.

[2] Goossens J., Richard P., 2004. Overview of real-time scheduling problems. *Euro Workshop on Project Management and Scheduling*

[3] Liu C. L., Layland J. W., 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, Vol. 20, No.1, pp 46-61.

[4] Hong J., Tan X., Towsley D., 1989. A Performance Analysis of Minimum Laxity and Earliest Deadline Scheduling in a Real-Time System. *IEEE Trans. on Comp.*, Vol. 38, No. 12**.**

[5] Sha, L. and Goodenough, J. B., 1990. Real-Time Scheduling Theory and Ada. *IEEE Computer*, Vol. 23, No. 4, pp. 53-62.

[6] Wang Lie-Xin, 1996. A course in fuzzy systems and control. *Prentice Hall, Paperback, Published*.

[7] Andersson B., Jonsson J., 2000. Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition.

Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA'00)

[8] Mamdani E.H., Assilian S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13,.

[9] Sugeno, M., 1985. Industrial applications of fuzzy control. *Elsevier Science Inc*., New York, NY.

[10] Jang, J.-S. R., 1993. ANFIS: Adaptive-Network-based Fuzzy Inference Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685.

[11] Lauzac S., Melhem R., Mosse D., 1998. Comparison of Global and Partitioning Schemes for Scheduling Rate Monotonic Tasks on a Multiprocessor. *Euromicro Workshopon RealTime Systems*.

[12] Sabeghi M., Naghibzadeh M., Taghavi T., 2005. 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)*.

[13] Sabeghi M., Naghibzadeh M., Taghavi T., 2006. 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).*

[14] Mahdi Hamzeh, Sied Mehdi Fakhraie, and Caro Lucas, Soft real-time fuzzy task scheduling for multiprocessor systems, International journal of intelligent technology Vol. 2 No. 4, 2007, pp 211-216

[15] Hajar Siar, Seyedeh Habibe Nabavi, Shahaboddin, Shamshirband, Static task scheduling in cooperative distributed systems based on soft computing techniques, Australian journal of basic and applied sciences, Vol. 4, No. 6, 2010, pp. 1518-1526.

