Int. J. Adv. Res. Sci. Technol. Volume 2, Issue 3, 2013,pp.160-165.

IJARST

International Journal of Advanced Research in Science and Technology

journal homepage: www.ijarst.com

ISSN 2319 – 1783 (Print)

ISSN 2320 – 1126 (Online)

Fuzzy Based Real Time Task Scheduling for Multiprocessor System

Satyajit Nayak and Pratyush Ranjan Mahapatra*

Gandhi institute for Technology, Bhubaneswar, Odisha, India

****Corresponding Author's Email: satyajitnayak@gift.edu.in***

A R T I C L E I N F O

Article history:

Received 03 Dec. 2013

Accepted 25 Dec. 2013

Available online 31 Dec. 2013

Keywords:

Fuzzy inference systems,
Fuzzy inference engine,
Model.

A B S T R A C T

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. Real-time 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.

Introduction:

There are a plenty of real-time scheduling algorithms that are proposed in the literature. Each of these algorithms bases its decision on certain parameter while attempting to schedule tasks to satisfy their time requirements. Some algorithms use parameters that are determined statically such as the Rate Monotonic algorithm that uses the request interval of each task as its priority [4, 7]. Others use parameters that are calculated at run time. Laxity and deadline are among those parameters that are the most considered. Laxity says the task execution must begin within a certain amount of time while deadline implies the time instant at which its execution must be completed.

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, regency 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 [27]. 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 [27]. First is called Mamdani's fuzzy inference method proposed in 1975 by Ebrahim Mamdani [28] and the other is Takagi-Sugeno-Kang, or simply Sugeno, method of fuzzy inference introduced in 1985 [29]. 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.

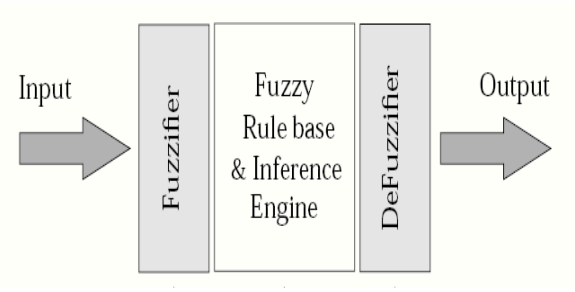

Fig. 1: Block diagram of fuzzy inference process

Fuzzy Algorithms:

MFDF, MFLF, FGEDF, FPEDF, FGMLF and FPMLF algorithms has been considered here. Each of the algorithms is as follows:

Algorithm MFDF:

Loop

1. For each task T, 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 (laxity, deadline, etc)

End loop

Loop

1. For each task T, feed its external priority and laxity into the inference engine. Consider the output of inference module as priority of task T.

The 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". An example of fuzzy IF-THEN rules is: IF frequency is high then priority is low, which frequency and priority are linguistics variables and high and low are linguistics terms. Typical fuzzy inference systems have dozens of rules. These rules are stored in a knowledgebase.

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 (laxity, deadline, etc)

End loop

The algorithm for the MFDF is similar to the MFLF with laxity replaced by deadline.

Algorithm FGEDF:

Loop

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 MFLF

Loop

1. For each task T, feed its external priority and laxity 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 (laxity, deadline, etc)

End loop

The algorithm for the MFDF is similar to the MFLF with laxity replaced by deadline.

Algorithm FGEDF

Loop

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 FGMLF:

Loop

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 laxity 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

FGMLF is much the same with FGEDF just by replacing the word deadline by laxity.

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

Algorithm FPMLF 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 laxity 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

FPMLF is much the same with FPEDF just by replacing the word deadline by laxity.

For a detail explanation of each algorithm refer to [2,3,9,12].

Scheduling Algorithms and Different Task Model:

Task Model:

A task is a complete sequence of instructions. Task execution starts when a task is selected by task dispatcher and one of the system's processors starts to run task's instructions. Tasks are classified according to their deadline, priority, arrival characteristic, and computation cycles requests.

Scheduling Algorithms:

First-Come-First-Served (FCFS) algorithm [20] selects the task with the earliest arrival time. If system contains periodic tasks, their release time will be considered. This algorithm makes no effort to consider a task's deadline.

Earliest Deadline First (EDF) algorithm [15, 20] always chooses the task with the earliest deadline. It has been proved that this algorithm is optimal in a uni-processor system. Since it cannot consider priority and therefore cannot analyze it, this algorithm fails under overloading conditions.

Fuzzy Inference Engine:

Fuzzy inference is the process of formulating the mapping from a given input set to an output using fuzzy logic. The basic elements of fuzzy logic are linguistic variables, fuzzy sets, and fuzzy rules [32]. The linguistic variables' values are words, specifically adjectives like "small," "little," "medium," "high," and so on. A fuzzy set is a collection of couples of elements. It generalizes the concept of a classical set, allowing its elements to have a partial membership. The degree to which the generic element "x" belongs to the fuzzy set A (expressed by the linguistic statement x is A) is characterized by a membership function (MF), fA(x). The membership function of a fuzzy set corresponds to the indicator function of the classical sets. It can be expressed in the form of a curve that defines how each point in the input space is mapped to a membership value or a degree of truth between 0 and 1. The most common shape of a membership function is triangular, although trapezoidal and bell curves are also used. This operation normalizes all inputs to the same range and has a direct effect on system performance and accuracy.

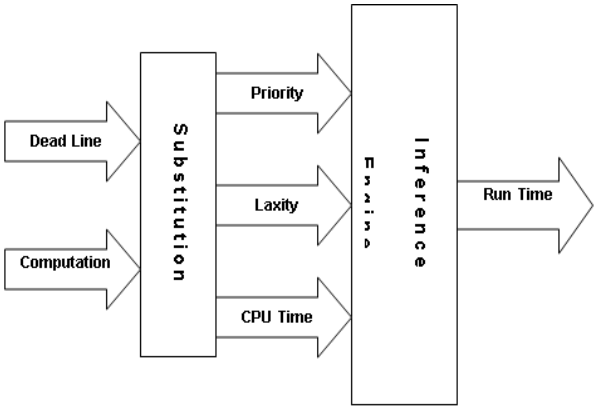

Fig. 2: fuzzy inference engine

A fuzzy set A is defined within a finite interval called universe of discourse U as follows:

$$A = \{(x, f(x)), f(x): U \rightarrow [0, 1]\}$$

U is the whole input range allowed for a given fuzzy linguistic variable. All fuzzy sets related to a given variable make up the term set, the set of labels within the linguistic variable described or, more

properly, granulated. Fuzzy rules form the basis of fuzzy reasoning. They describe relationships among imprecise, qualitative, linguistic expressions of the system's input and output. Generally, these rules are natural language representations of human or expert knowledge and provide an easily understood knowledge representation scheme. A typical conditional fuzzy rule assumes a form such as

IF Speed is "Low" AND Race is "Dry" THEN Braking is "Soft".

Proposed Model & Proposed Algorithm:

Proposed algorithm:

1. For each task of input queue
a. Feeds task's run-time priority using fuzzy inference engine
2. While system has a free processor
a. Assign the task with highest run-time priority to the processor
3. Loop forever
a. If processor event occurs
i. Go to 2.
b. If scheduling event occurs
i. Update tasks parameters.
ii. Go to 1.

Satisfactory performance is achieved by using 39 Sugeno rules only. This number is obtained by simplifying 169 rules in different examples. Some of them are mentioned below:

• If (Laxity is "Very low") and (Priority is "Very high") then
R Priority =100× priority −10×laxity.
• If (Laxity is "Low") and (Priority is "Very high") then
R Priority = 50× priority − 20×laxity.
• If (Laxity is "Medium") and (Priority is "Normal") and
(CPU time is "High") then
R Priority = 25 x priority - 40 x laxity – 50 x CPUtime

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.

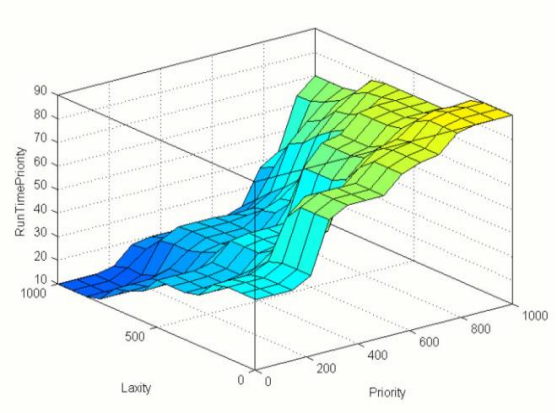

Fig. 3: The decision surface corresponding to inference rules.

Experimental Results:

Performance metrics, which are used to compare different algorithms, must be carefully chosen to reflect the real characteristics of a system. These metrics are as follows. Response time, which is defined as the amount of time a system takes to react to a given input, is one of the most important factors in most scheduling algorithms. Number of missed deadlines is an influential metric in scheduling algorithms for soft real-time systems. When task preemption is allowed, another prominent metric comes into existence and that is the number of preemptions. Each of preemptions requires the system to perform a context switching which is a time consuming action. CPU utilization is also an important metric because the main goal of a scheduling algorithm is to assign and manage system resources so that a good utilization is achieved. Yet another metric, which is considered in our study, is the number of missed deadlines from the class of highest priority tasks. This corresponds to the external priority being *very high*.

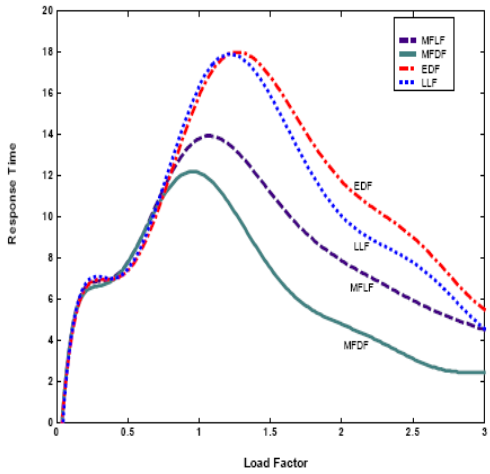

Fig. 4: Response time in overloaded conditions

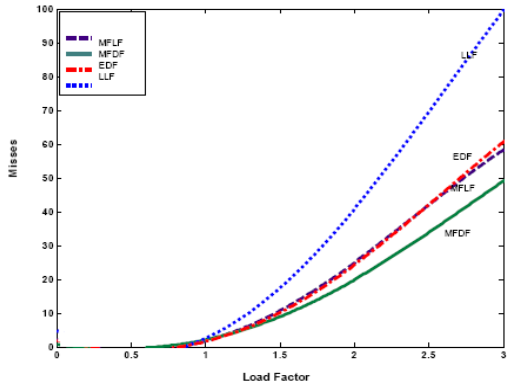

Fig. 5: Number of Misses

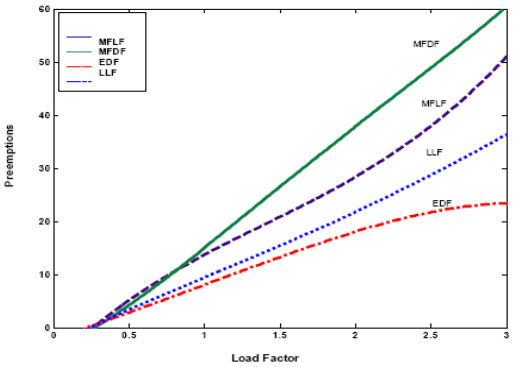

Fig. 6: Number of Preemptions

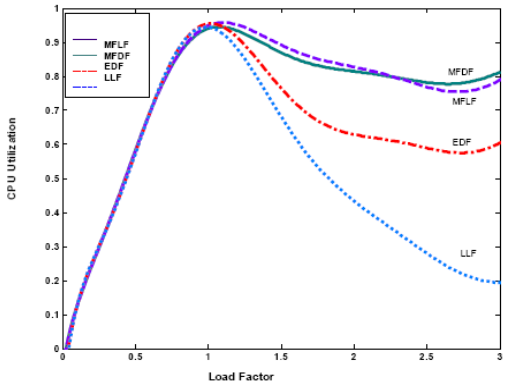

Fig. 7: CPU Utilization

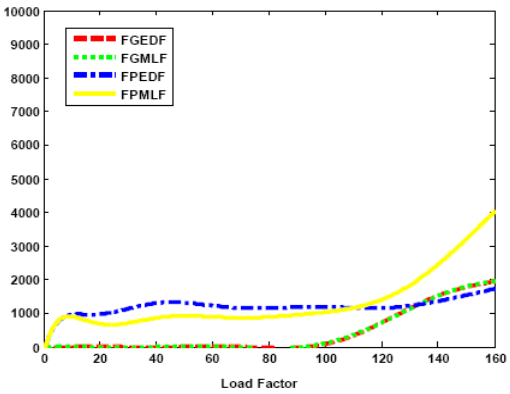

Fig. 8: Number of misses for 128 CPUs

500 x 500	**17172**	**17203**	**17093**	**8016**

Note: Units are in terms of Ms

Conclusion:

This thesis compared fuzzy algorithms based on deadline and laxity. As it was shown, using deadline as a fuzzy parameter in real-time scheduling is more promising than laxity. Also, it seems that partitioning approach almost outperforms global approach for multiprocessor real-time scheduling.

The proposed scheduler which proposed in this thesis has low complexity due to the simplicity of fuzzy inference engine. As a consequence, its computation complexity and response time is constant and by increasing the number of processors will not increase. This model is efficient when system has heterogeneous tasks with different constraints.

Our future work is to map this algorithm on our real-time fuzzy processor.

References:

1. Liu C. L., Layland J. W., Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM, 20(1):46-61, 1973.
2. F. Gruian, "Energy-centric scheduling for real-time systems," in Department of Computer Science. PhD dissertation: Lund University, 2002, p. 164.
3. Z. Deng, J. W. Liu, and S. Sun, "Dynamic scheduling of hard realtime Applications in open system environment," Tech. Rep., University of Illinois at Urbana-Champaign 1996.
4. G. Buttazzo and J. A. Stankovic, "RED: robust earliest deadline Scheduling," in Proc. 3rd Intl. Workshop Responsive Computing Systems, Lincoln, NH, 1993, pp. 100-111.
5. J. Zhu, T. G. Lewis, W. Jackson, and R. L. Wilson, "Scheduling in hard real-time applications," IEEE Soft., vol. 12, pp. 54-63, 1995.
6. K. Taewoong, S. Heonshik, and C. Naehyuck, "Scheduling algorithm for hard real-time communication in demand priority network," in Proc. 10th Euromicro Workshop Real-Time Systems, Berlin, Germany, 1998, pp. 45-52.
7. C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," J. ACM, vol. 20 pp. 46-61, 1973.
8. D. Babbar and P. Krueger, "On-line hard real-time scheduling of parallel tasks on partitionable multiprocessors," in Proc. Intl. Conf. Parallel Processing, 1994, pp. 29-38.
9. 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, pp. 4022-4027 vol.5.
10. T. F. Abdelzaher and K. G. Shin, "Comment on a pre-run-time scheduling algorithm for hard real-time systems," IEEE Trans Software Engineering, vol. 23, pp. 599-600, Sep 1997.
11. K. Ramamritham and J. A. Stankovic, "Dynamic task scheduling in hard real-time distributed systems," IEEE Softw., vol. 1, pp. 65-75, July 1984.
12. P. A. Laplante, "The certainty of uncertainty in real-time

systems," IEEE Instrum. Meas. Mag., vol. 7, pp. 44-50, Dec 2004.
13. K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," Proc. IEEE, vol. 82, pp. 55-67, Jan 1994.
14. 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, pp. 155-159.
15. N. D. Thai, "Real-time scheduling in distributed systems," in Proc. Intl. Conf. Parallel Computing in Electrical Engineering, Warsaw, Poland, 2002, pp. 165-170.
16. C. Lin and S. A. Brandt, "Efficient soft real-time processing in an integrated system," in Proc. 25th IEEE Real-Time Systems Symp., 2004.
17. I. E. W. Giering and T. P. Baker, "A tool for the deterministic scheduling of real-time programs implemented as periodic Ada tasks," Ada Lett., vol. XIV, pp. 54-73, 1994.
18. L. Hluchý, M. Dobrucký, and J. Astalos, "Hybrid approach to task allocation in distributed systems," in Proc. 4th Intl. Conf. Parallel Computing Technologies, Yaroslavl, Russia, 1997 pp. 210-215.
19. G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," IEEE Trans. Comput., vol. 51, pp. 289-302, Mar 2002.
20. A. S. Tanenbaum, Distributed operating systems: Prentice Hall, 1994.
21. J. Lee, A. Tiao, and J. Yen, "A fuzzy rule-based approach to real-time scheduling," in Proc. 3rd IEEE Conf. Fuzzy Systems, IEEE World Congress Computational Intelligence, FL, 1994, pp. 1394-1399 vol.2.
22. M. Silly-Chetto, "Dynamic acceptance of a periodic tasks with periodic tasks under resource sharing constraints," IEE Proc. Software, vol. 146, pp. 120-127, Apr 1999.
23. M. Caccamo and G. Buttazzo, "Exploiting skips in periodic tasks for enhancing aperiodic responsiveness," in Proc. 18th IEEE Real-Time Systems Symp., San Francisco, CA, 1997, p. 330.
24. J. Mario J. Gonzalez, "Deterministic processor scheduling," ACM Comput.Surv., vol. 9, pp. 173-204, 1977.
25. 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, pp. 164-173.
26. 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, pp. 1-10.
27. Wang Lie-Xin, 1996. A course in fuzzy systems and control. *Prentice Hall, Paperback, Published*.
28. 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,.
29. Sugeno, M., 1985. Industrial applications of fuzzy control. *Elsevier Science Inc*., New York, NY.
30. L. A. Zadeh, "Fuzzy sets versus probability," Proc. IEEE, vol. 68, pp. 421-421, March 1980.
31. L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing,"Commun. ACM, vol. 37, pp. 77-84, March 1994.
32. W. Pedrycz and F. Gomide, An introduction to fuzzy sets: analysis and design: The MIT Press, 1998.
33. E. H. Mamdani, "Application of fuzzy algorithms for the control of adynamic plant," Proc. IEE, vol. 121, pp. 1585-1588, Dec 1974.
34. T. Takagi and M.Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. Syst., Man,
35. Cybern., vol. 15, pp. 116-132, 1985.
36. H. Surmann and A. P. Ungering, "Fuzzy rule-based systems on general-purpose processors," IEEE Micro, vol. 15, pp. 40-48, Aug 1995.
37. G. Ascia and V. Catania, "A general purpose processor oriented to fuzzy reasoning," in Proc. 10th IEEE International Conf. Fuzzy Systems, Melbourne, Australia, 2001, pp. 352-355.
38. K. Youngdal and L.-K. Hyung, "An architecture of fuzzy logic controller with parallel defuzzification," in Proc. Biennial Conf. of the North American Fuzzy Information Processing Society, Berkeley, CA, 1996, pp. 497-501.