Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic

[1]Bashir Alam, [1]M.N. Doja, [2]R. Biswas

[1]*Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India-110025*

[2]*Department of Computer Science and Engineering, ITM, Gurgaon, Haryana, India -122017*

Email: - babashiralam@gmail.com, ndoja@yahoo.com , ranjitbiswas@yahoo.com

Abstract

In Round Robin Scheduling the time quantum is fixed and then processes are scheduled such that no process get CPU time more than one time quantum in one go. If time quantum is too large, the response time of the processes is too much which may not be tolerated in interactive environment. If time quantum is too small, it causes unnecessarily frequent context switch leading to more overheads resulting in less throughput. In this paper a method using fuzzy logic has been proposed that decides a value that is neither too large nor too small such that every process has got reasonable response time and the throughput of the system is not decreased due to unnecessarily context switches.

Keywords: - CPU Scheduling, Round Robin, FIS

1. Introduction

When a computer is multiprogrammed, it frequently has multiple processes competing for the CPU at the same time. When more than one process is in the ready state and there is only one CPU available, the operating system must decide which process to run first. The part of operating system that makes the choice is called short term scheduler or CPU scheduler. The algorithm that it uses is called scheduling algorithm. There are several scheduling algorithms. Different scheduling algorithms have different properties and the choice of a particular algorithm may favor one class of processes over another. Many criteria have been suggested for comparing CPU scheduling algorithms and deciding which one is the best algorithm [1]. Some of the criteria include (i)Fairness(i)CPU utilization(iii)Throughput (iv)Turnaround time (v)Waiting time (vi)Response time. It is desirable to maximize CPU utilization and throughput, to minimize turnaround time, waiting time and response time and to avoid starvation of any process.[1,2] Some of the scheduling algorithms are briefly described below: **FCFS**: In First come First serve scheduling algorithm the process that request first is scheduled for execution[1,2,3]**SJF**: In shortest Job first scheduling algorithm the process with the minimum burst time is scheduled for execution.[1,2]**SRTN**: In shortest Remaining time next scheduling algorithm , the process with shortest remaining time is scheduled for execution.[3]**Priority**: in Priority Scheduling algorithm the process with highest priority is

978-0-7695-3504-3/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCEE.2008.89

scheduled for execution.[1, 2, 3]**Multilevel queue scheduling**: In this the ready queue is partitioned into several separate queues. The processes are permanently assigned to one queue generally based on some property of the process such as memory size, process priority or process type. Each queue has its own scheduling algorithm. There is scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. Each queue has absolute priority over low priority queues. [1]**Multilevel feedback-queue scheduling**:-This is like **Multilevel queue scheduling** but allows a process to move between queues.[1] **Fair share Scheduling**: Fair share scheduler considers the execution history of a related group of processes, along with the individual execution history of each process in making scheduling decision. The user community is divided into a fair- share groups. Each group is allocated a fraction of CPU time. Scheduling is done on the basis of priority of the process, Its recent processor usage and the recent processor usages of the group to which the process belongs. Each process is assigned a base priority. The priority of a process drops as the process uses the processor and as the group to which process belongs uses the processor.[3]**Guaranteed scheduling**:-In this a ratio of actual CPU time a process had and its entitled CPU time is calculated. The process with this lowest ration is scheduled[2] Lottery Scheduling:-The basic idea is to give processes lottery tickets for CPU time. Whenever a scheduling decision has to be made , a lottery ticket is chosen at random and the process holding the ticket gets the CPU.[2] **HRRN**: - In this response ratio is calculated for each process. The process with the highest ratio is scheduled for execution. [3] **Round-robin**: In this the CPU scheduler goes around the ready queue allocating the CPU to each process for a time interval of up to one time quantum. If time quantum is too large, the response time of the processes is too much which may not be tolerated in interactive environment. If time quantum is too small, it causes unnecessarily frequent context switch leading to more overheads resulting in less throughput. In this paper a method using fuzzy logic has been proposed that decides a value that is neither too large nor too small such that every process has got reasonable response time and the throughput of the system is not decreased due to unnecessarily context switches.

2. Fuzzy Inference Systems and Fuzzy Logic

A fuzzy inference system (FIS) tries to derive answers from a knowledgebase by using a fuzzy inference engine. The inference engine which is considered to be the brain of the expert systems provides the methodologies for reasoning around the information in the knowledgebase and formulating the results. Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth that 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 the 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. 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. The input space is sometimes referred to as the universe of discourse [4]. Fuzzy Inference Systems are conceptually very simple. An FIS consists of an input stage, a processing stage, and an output stage. The input stage maps the inputs, such as deadline, execution time, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a result for each. It then combines the results of the rules. Finally, the output stage converts the combined result back into a specific output value [4]. As discussed earlier, the processing stage, which is called 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". Typical fuzzy inference subsystems have dozens of rules. These rules are stored in a knowledgebase. An example of fuzzy IF-THEN rules is: IF *number of users* is *high* then *time quantum* is *low*, in which *number of user* and *time quantum* are linguistics variables and *high* and *low* are linguistics terms. The five steps toward a fuzzy inference are as follows:

(i) fuzzifying inputs(ii) applying fuzzy operators(iii)applying implication methods (iv)aggregating outputs(v) defuzzifying results

Bellow is a quick review of these steps. However, a detailed study is not in the scope of this paper. Fuzzifying the inputs is the act of determining the degree to which they belong to each of the appropriate fuzzy sets via membership functions. Once the inputs have been fuzzified, the degree to which each part of the antecedent has been satisfied for each rule is known. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one value that represents the result of the antecedent for that rule. The implication function then modifies that output fuzzy set to the degree specified by the antecedent. Since decisions are based on the testing of all of the rules in the Fuzzy Inference Subsystem (FIS), the results from each rule must be combined in order to make the final decision. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are processed into a single fuzzy set. The input for the defuzzification process is the aggregated output fuzzy set and the output is then a single crisp value [4]. This can be summarized as follows: mapping input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single crisp valued output. There are two common inference methods [4]. The first one is called Mamdani's fuzzy inference method proposed in 1975 by Ebrahim Mamdani [5] and the second one is Takagi-Sugeno-Kang, or simply Sugeno, method of fuzzy inference introduced in 1985 [6]. 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's output membership functions are either linear or constant but Mamdani's inference expects the output membership functions to be fuzzy sets. Sugeno's method has three advantages. Firstly, it is computationally efficient, which is an essential benefit to real-time systems. Secondly, it works well with optimization and adaptive techniques. These adaptive techniques provide a method for the fuzzy modeling procedure to extract proper knowledge about a data set, in order to compute the membership function parameters that

best allow the associated fuzzy inference system to track the given input/output data. The third, advantage of Sugeno type inference is that it is well-suited to mathematical analysis.

3. FIS for finding time quantum

The Fuzzy Inference System for finding the time quantum has got 2 inputs and one output. First input is N that specify the number of user/ processes in the system and second input is the average burst time of the processes in the ready queue. Time quantum is the output of the FIS.

Membership Function for N

Type- Triangular, Range:1-10, low-[0,2,4], medium-[3,5.5,8] High[7,8.5,10]

Membership Function the Average Burst Time

Type- Triangular, Range:0-10, low-[-4,0,4], medium-[3,5,7] High:-[6,10,16]

Membership Function for Time Quantum

Type- Triangular, Range:1-5, low-[0,1,2], medium-[1,2.5,4] High :-[3,5,7]

RULE BASE FOR FIS

S.No.	N	ABT	Time Quantum
1	low	low	low
2.	low	medium	medium
3.	low	high	high
4.	medium	low	medium
5.	medium	medium	medium
6.	medium	high	medium
7.	high	low	low
8.	high	medium	low
9.	high	high	medium

4. Proposed Algorithm

Step I- Find ABT, the average burst time of the processes

Step II- Give N,the number of users and ABT to the FIS designed above.

StepIII- Take output of FIS as the time quantum

Step IV- Invoke Round Robin Scheduling Algorithm

5. onclusion

In this paper we have designed a FIS that compute a suitable time quantum for a given CPU scheduling scenario.

References

1] Silberschatz, A., Peterson, J. **L.,** and Galvin, P.B.,***Operating System Concepts,*** Addison Wesley, 7th Edition, **2006.**

[2]Andrew S. Tanenbaum , and Albert S. Woodfhull , Operating Systems Design and Implementation,Second Edition,2005

[3]William Stallings, Operating Systems Internal and Design Principles, 5th Edition ,2006

[4]Wang Lie-Xin, A course in fuzzy systems and control, Prentice Hall, August 1996.

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

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

[7]Jang, J.-S. R., ANFIS: Adaptive-Network-based Fuzzy Inference Systems, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23(3), 685, May 1993.

[8] Simon D, Training fuzzy systems with the extended Kalman filter, Fuzzy Sets and Systems, Vol. 132, No. 2, 1, December 2002.