Fuzzy Based Multilevel Queue Scheduling Algorithm

Vaishali Chahar
CSE Department, ITM University
Gurgaon, India
tn.chahar@gmail.com

Supriya Raheja
CSE Department, ITM University
Gurgaon, India
supriya@itmindia.edu

Abstract— **Many scheduling techniques has been designed for scheduling of processes in the multiprogramming system, one of these techniques is multilevel queue CPU scheduling technique where the ready queue is divided into multiple sub-queues. Processes are assigned to different sub queues depending on their classification. In traditional multilevel queue scheduling approach each queue is assigned a priority level. Processes from the higher level queues always gets priority over the processes in lower level queues which can cause starvation for processes in lower queues. In this paper we propose a new fuzzy based multilevel queue CPU scheduling algorithm. In our proposed work ready queue is divided into two sub-queues which contain I/O bound and CPU bound processes. CPU time is allocated dynamically to each queue. We have designed two fuzzy inference systems. One is to dynamically allocate the CPU time to two queues. Another FIS is designed to calculate the time quantum to schedule the I/O bound processes. Our proposed work improves the starvation problem and the average response time.**

Keywords: Fuzzy Inference System (FIS); CPU Scheduling; CPU Bound, I/O Bound; First Come First Served (FCFS); Round Robin (RR).

I. INTRODUCTION

Multiprogramming system allows multiple jobs to be executed concurrently, by switching the CPU time among multiple processes existing in the system. To schedule multiple processes inside the system for execution scheduler is required. Whenever the CPU becomes idle or completes execution of a process, CPU scheduler selects another process from the ready queue to run next. CPU scheduler requires the scheduling algorithm to choose next process to run. There are many scheduling algorithms, each with its own characteristics.

Multilevel Queue CPU scheduling algorithm is one of the algorithms which we will discuss in this paper.

A. Different scheduling techniques

First come first served: In this scheduling technique [1, 2] the jobs in ready queue are scheduled in the order in which they appear in the queue. Job which comes first gets CPU time first then next and so on. It is a traditional scheduling technique where all the jobs have same priority.

Round Robin Scheduling: In round robin CPU scheduling technique [1, 2] multiple processes residing in ready queue are scheduled using a time sharing format where each process gets equal amount of CPU time. The time or the quantum of time for which the processes gets CPU can be a fixed time quantum calculated by user or can be a variable which can be calculated at variable time. After the time quantum allocated to a process expires, CPU is allocated to the next process in the queue. This Scheduling technique has no starvation problem.

Shortest Job first: The Job in the queue which has the shortest burst time is scheduled first to run [1] and then the next shortest job gets the CPU time. In this scheduling technique if there are large number of small burst time processes exist in the system then processes having larger burst time have to wait for a long time to get the CPU.

Priority Scheduling: In this scheduling technique [1, 2, 11] processes are assigned a priority value where the highest priority process always gets CPU first for execution; priority can be assigned depending upon various factors such as type of process, behaviour of process etc.

Fair share scheduling: In fair share scheduling algorithm [3] all the processes gets fair share of CPU resource. No process has higher priority than others; all are of same level and gets same amount of CPU share.

Multilevel Queue Scheduling: In this scheduling technique [1, 2] ready queue is divided into multiple sub-queues and processes are assigned to theses queues depending on their classification which further depend upon process characteristics. Each sub queue in multilevel queue scheduling technique has a priority level assigned to them. Queue containing high priority processes are always scheduled first for execution. Processes in lower priority queues get CPU time only after the completion of processes in higher queues. Processes inside different queues are scheduled using different scheduling techniques such as RR, FCFS etc.

Multilevel feedback queue scheduling: In multilevel feedback queue scheduling [1, 2, 10] also ready queue is divided into multiple sub queues but in multilevel feedback queue scheduling technique processes can move in between multiple queues.

B. Different criteria used for comparing different scheduling algorithms are:

There can be many factors [1, 2, 4, 11] which should be taken care of while working for a CPU scheduler design. Values for

978-1-4673-6217-7/13/$31.00 ©2013 IEEE

some of the factors should be maximized and for others their values should be decreased.

I. *CPU Utilization*: keep CPU utilization as high as possible.
II. *Throughput*: number of tasks completed per unit time.
III. *Turnaround Time:* mean time from submission to completion of task.
IV. *Waiting Time:* amount of time spent ready to run but not running.
V. *Response Time:* time between submissions of requests to the first response to the request.
VI. *Loss ratio:* is the fraction of requests that are dropped due to their deadline cannot be met.

In proposed algorithm, we have improved the Multilevel Queue CPU scheduling technique using fuzzy logic. Section II includes the related work. In section III, we have discussed briefly the fuzzy inference engine. In section IV and V, we introduced a new methodology and new MFQ algorithm. Finally, Results have discussed in section VI.

II. Related Work

In existing multilevel queue scheduling technique[1] CPU time is time sliced statically between multiple queues where higher level queues always gets static, maximum share of CPU time and are always scheduled first for execution. Where processes in lower level queues are scheduled when the CPU time allocated to higher queues expires or all processes in higher queues completes their execution. This causes high response time and starvation problem for processes in lower level queues.

Processes inside different queues are scheduled using different scheduling techniques. Processes in high level queues are usually scheduled using round robin scheduling technique [5, 9], where processes residing in lower level queues is usually scheduled using the traditional FCFS [2] scheduling technique.

We have proposed a new fuzzy based multilevel queue CPU scheduling algorithm, where dynamic CPU allocation strategy is being used instead of static CPU allocation to the queues. For the designing of FIS, we have used the fuzzy tool box [6]. We have designed a FIS to calculate dynamic CPU allocation and another to allocate the time quantum for Round Robin scheduling technique.

III. Fuzzy Inference System and Fuzzy Logic

Fuzzy inference system is the system that derives the conclusion to problems that are vague in nature using the fuzzy inference engine. Fuzzy inference engine takes the input and drives the output by using the rules [7] defined by the user; fuzzy rules are defined using IF-Then form. Fuzzy logic is an extension of Boolean logic [5, 8]; it deals with partial truth that denotes the extent to which a proposition is true. Boolean logic [8] deals with binary form where either a value can be related or not related to a set i.e. either 0 or 1.

Partial truth in fuzzy logic is defined by the membership function which corresponds to the characteristic function of the classical sets. It can be expressed in the form of a graph that maps each point to a membership value or a degree of truth between 0 and 1. The most commonly used shape of a membership function is triangular and trapezoidal [5, 7]. FIS usually consists of an input stage, a processing stage, and an output stage. There are basically two common types of inference engine [7]. The first one is called Mamdani's engine proposed in 1975 by Ebrahim Mamdani and the second one is Takagi-Sugeno-Kang fuzzy inference engine introduced in 1985. The main difference between Mamdani and Sugeno is in their outputs. In Sugeno's output membership functions are either linear or constant but in Mamdani's inference engine the output membership functions are fuzzy sets. However the detailed study of these two is not in the scope of the paper.

In proposed work we have designed two mamdani style inference engines using the matlab fuzzy toolbox [6]. In proposed technique fuzzy inference systems are designed to calculate the values dynamically. Fuzzy inference systems works best when required to calculate new values every time, as they work on the fuzzy inference rules which are decide by the users itself and can be modified whenever required.

IV. Proposed Methodology

A new multilevel queue scheduling algorithm using fuzzy logic has been proposed; where dynamic CPU allocation strategy is being used i.e. the CPU time is allocated dynamically among different queues using a fuzzy inference system.

An inference engine DA_FIS shown in fig.2 is designed, which decides what percentage of CPU time to be allocated to different queues in each cycle. In proposed work cycle time is decided by the user and cycle time represents the total time during which CPU is available to different queues. The value of CPU time allocated to different queues is recalculated after each cycle.

In proposed technique ready queue is divided into two sub queues Q1 and Q2 as shown in Fig.1 where I/O bound processes are scheduled in queue Q1 and CPU bound processes are scheduled in queue Q2.

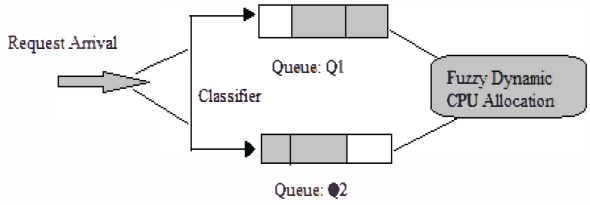


Fig.1 Schematic View of Proposed System

Queue Q1 gets preference over the queue Q2 for CPU time. Q2 gets the CPU share only after the expiry of CPU time allocated to queue Q1 or if all the processes of queue Q1 completes their execution. CPU time is allocated to both the queues using the designed fuzzy inference system DA_FIS. Inside Q1, processes are scheduled using RR scheduling algorithm. Another FIS QA_FIS has been designed to calculate the value of time quantum as shown in fig 4. Inside Q2, processes are scheduled using FCFS scheduling algorithm.

CPU bound and I/O bound processes are classified depending upon their required CPU time and I/O operation time. I/O bound processes are those which require small CPU time slice and which perform more I/O operation, they are also known as interactive processes which frequently require input/output from the users. CPU bound processes are those which do not interact much with user but they require maximum CPU time to complete their execution.

A. PROPOSED FIS FOR CALCULATION OF CPU TIME

Designed inference engine DA_FIS calculates the amount of CPU time to be allocated to the queue Q1, based upon two parameters: No of processes in the queue (NP_Q) and User's priority (UP) assigned by the user as shown in Fig 2. User's priority ranges from 1-10 where 1 represents the lower priority and 10 represent the highest priority.

The CPU time allocated to queue Q2 is calculate by subtracting the amount of CPU time allocated to queue Q1 from total amount of time represented by the cycle time. Allocation of CPU time is recalculated after each cycle. In our proposed work cycle time is kept constant and is decided by the user.

The designed FIS is as follows:

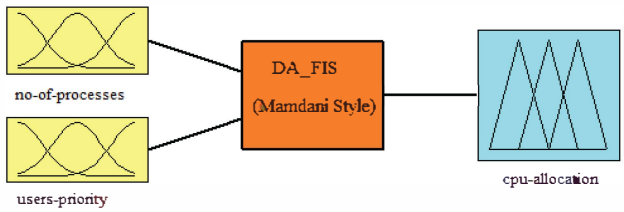


Fig.2 Proposed Fuzzy Inference System (DA_FIS)

For our proposed inference engine DA_FIS, we have defined total nine rules as shown in fig.3.

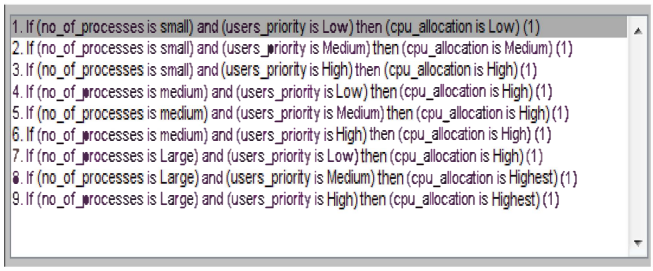


Fig.3 Fuzzy Rules for DA_FIS

B. FIS FOR RR SCHEDULING

The I/O bound processes inside the queue Q1 are scheduled in round robin fashion. Initially all the processes in the queue Q1 are sorted in increasing order of their burst time, processes having shortest burst time are scheduled first for execution.

A mamdani style inference engine QA_FIS shown in fig.4 has designed for calculation of the value of time quantum. The value of time quantum (CTQ) depends upon two values: No of processes in the queue (NP) and average burst time (ABT) of the processes. After each cycle new value of time quantum will be calculated. After each cycle new value of time quantum is calculated.

In the proposed round robin technique if the time quantum assigned to a process expires and its remaining execution time is one or less than one, then CPU will again be allocated to same process instead of moving to next process scheduled. This decreases the number of context switch required.

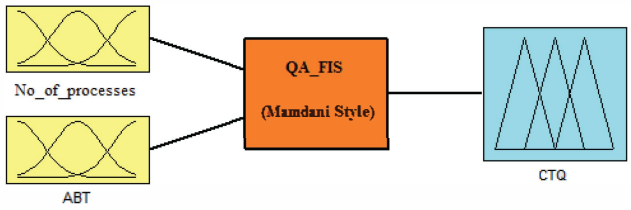


Fig.4 Proposed Fuzzy Inference for RR Scheduling

For the designed inference system QA_FIS we have designed a total of nine fuzzy rules as shown below in fig.5.

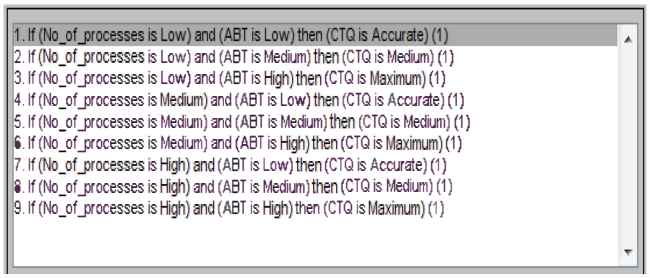


Fig.5 Rule Base for RR Scheduling

Scheduling of processes inside queue Q2 containing CPU bound processes has been done using FCFS technique with variation that all processes are scheduled to run for the first time as soon as they arrive in the queue so that their response time is improved.

V. PROPOSED ALGORITHM

Step1. Classify Processes into CPU bound & I/O bound.

Step2. Assign I/O bound processes to Q1 and CPU bound processes to Q2.

Step3. Calculate the percentage of CPU time allocation for Q1 using designed DA_FIS.

Step4. Calculate the CPU time assigned to Q2 by subtracting time assigned to Q1 from total percentage i.e. 100.

Step5. Arrange processes in queue Q1 so that process having shortest burst time gets CPU time first.

Step6. Processes in queue Q1 are scheduled using RR scheduling technique, value of time quantum is calculated for each cycle using proposed inference engine QA_FIS.

Step7. Schedule processes to run in queue Q2 when the CPU time allocated to queue Q1 expires or all processes in Q1 completes their execution.

Step8. Schedule processes in queue Q2 using modified FCFS technique discussed in section IV.

Step9. After each cycle go to step1.

VI. RESULTS

In this paper we compared our technique the already existing technique [1] discussed in related work in section III and the proposed multilevel queue scheduling technique. To validate the performance average response time and waiting time of processes has compared. Results shows, using proposed technique average response time has decreased. Additionally the starvation problem has also improved with slight increase in average waiting time.

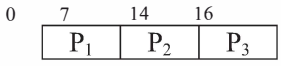
In given example we have considered 16 processes where 11 are I/O bound and 5 are CPU bound processes shown in table 1. Assume all time measures are in seconds.

Table I. Scheduling Example

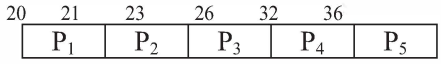
I/O Bound Processes	Arrival Time	Burst Time	CPU bound Processes	Arrival Time	Burst Time
P_1	0	8	C_1	0	20
P_2	0	9	C_2	0	25
P_3	0	5	C_3	30	30
P_4	21	6	C_4	35	35
P_5	25	7	C_5	40	15
P_6	38	4			
P_7	40	3			
P_8	45	5			
P_9	55	6			
P_{10}	57	5			
P_{11}	60	4			

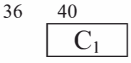
With the existing technique, the processes in Q1 are scheduled according to RR technique and the value of time quantum is fixed. Q2 processes are scheduled using FCFS technique. The constant value of a cycle is taken as 20 seconds.

1st Cycle

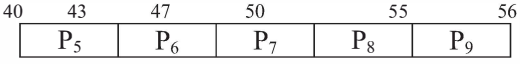


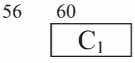
2nd cycle



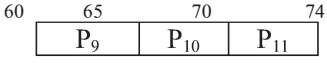


3rd cycle

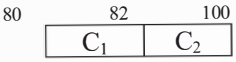




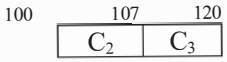
4th cycle



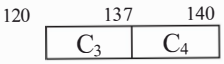
5th cycle



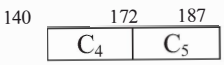
6th cycle



7th cycle



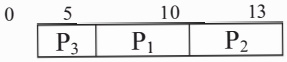
8th, 9th cycle

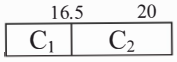


Average waiting time in this existing technique is 34.8sec where the response time is 29.8 sec.

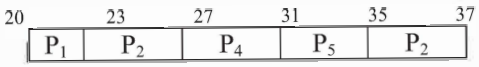
The same example is applied to the proposed technique. The value of CPU time allocated to each queue is calculated after each cycle. Moreover the value of time quantum varies dynamically in each cycle. Cycle time is taken as 20 sec.

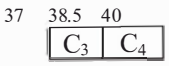
1st cycle
Calculated burst time (BT) is 5, CPU allocation to Q1 is 65%.



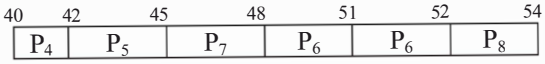


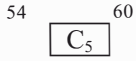
2nd cycle
Calculated burst time (BT) is 4, CPU allocation to Q1 is 85%.



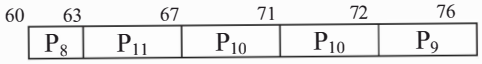


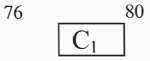
3rd cycle
Calculated burst time (BT) is 3, CPU allocation to Q1 is 70%.



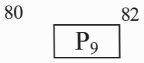


4th cycle
Calculated burst time (BT) is 4, CPU allocation to Q1 is 80%.

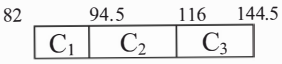


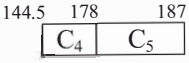


5th cycle
Calculated burst time (BT) is 2, CPU allocation to Q1 is 15%.



6th cycle
Queue Q1 is empty so entire CPU time cycle i.e. 20 sec is given to Q2.





Average waiting time in case of proposed technique is 38 sec and calculated average response time is 8.12 sec. Comparison of waiting and response time of two techniques has been shown in fig.6 and fig.7 respectively.

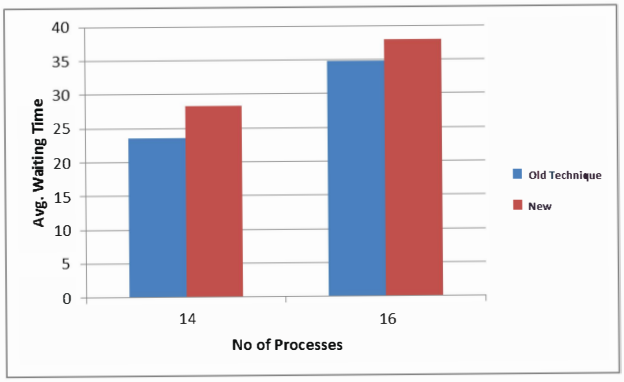


Fig.6 Comparison of Average Waiting Time

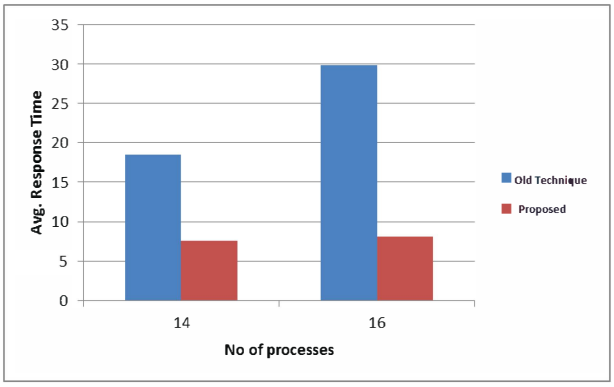


Fig.7 Comparison of Average Response Time

VII. CONCLUSION

In this paper we proposed a new fuzzy based Multilevel Queue CPU scheduling algorithm. In proposed algorithm CPU time is allocated dynamically among the two queues and also the value of time quantum is calculated variably. Fuzzy inference systems are used to calculate the values dynamically.

We implemented the algorithm using the fuzzy tool box provided by Matlab. We have compared the proposed algorithm with the existing multilevel queue scheduling technique and it is concluded that proposed algorithm improves the response time. Additionally it improves the starvation problem of processes but with slight increase in waiting time which can be ignored as the average reduction in waiting time is more as compared to the average increase in waiting time.

Our future work is focused on the further reduction in response and waiting time of processes when scheduled using multilevel queue scheduling technique.

REFERENCES

[1] Silberschatz and Galvin, Gagne, Operating system concepts, 6th ed., wiley india, 2008, pp. 151-188.

[2] Stallings, W. 2006, Operating Systems Internal and Design Principles. 5th Edition, Pearson Education.

[3] Fair Share Scheduling [Online]. Available: http://en.wikipedia.org/wiki/Fair-share_scheduling. Last Access -4/2013.

[4] Scheduling Criteria [Online]. Available: http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node120.html. Last access-04/2013.

[5] B. Alam and M.N. Doja, R. Biswas, "Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic", International Conference on Computer and Electrical Engineering, Phuket, Thailand, 2008,pp. 795-798.

[6] J.S.Roger Jang and N. Gulley, Matlab Fuzzy logic toolbox, http://faculty.petra.ac.id/resmana/private/matlabhelp/pdf_doc/fuzzy/fuzzy_tb.pd , Version2.

[7] S.N Sivanandam and S.N.Deepa, Pronciples of Soft Computing, 1st ed., Wiley India, 2008, pp-313-474

[8] Boolean Logic [Online], Available: http://en.wikipedia.org/wiki/Boolean_logic , Last Access-04/2013.

[9] R. J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences, 2009, pp. 1831-1837.

[10] M. V. Panduranga Rao and K. C. Shet, "Analysis of New Multi-Level Feedback Queue Scheduler for Real Time Kernel", International Journal of Computational Cognition Vol. 8, 2010, pp. 5-16.

[11] S. J. Kadhim and K. M. Al-Aubidy, "Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm",Springer-Verlag Berlin Heidelberg, 2010, pp. 45-52.

[12] A. Rezaee and A. Masoud Rahmani et al.," A Fuzzy Algorithm for Adaptive Multilevel Queue Management with QoS Feedback", IEEE,2011, pp. 121-127.