Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review of Multilevel Queue and Multilevel Feedback Queue Scheduling Techniques

Vaishali Chahar
CSE Deptt,ITM University, Gurgaon
India

Supriya Raheja
CSE Deptt,ITM University,Gurgaon
India

Abstract— Multilevel Queuing and Multilevel Feedback Queuing is common in the CPU scheduling techniques used in operating systems. These techniques are common but still have some issues and have a wide scope of improvement. Very less literature review is present on these scheduling techniques. So in this paper different methods for scheduling these techniques proposed by different authors has been collected and discussed. Firstly an introduction to basic multilevel queue and multilevel feedback queue scheduling techniques are discussed. And after that a review of techniques proposed by different authors are discussed.

Keywords— MLQ, MLFQ scheduling, QoS, FCFS, HRRN

I. INTRODUCTION

CPU is one of the most important resources which require scheduling, on which the working and speed of a system depends. Scheduling of CPU resource has many ways by which it can be scheduled like FCFS, Round Robin, SJF, Priority Scheduling. But Scheduling a CPU which has different types of processes, which are required to run, can be scheduled using multilevel queue and multilevel feedback CPU scheduling techniques.
Multilevel queue (MLQ) and multilevel feedback queues (MLFQ) CPU scheduling are very complex scheduling techniques itself which make use of basic CPU scheduling techniques available. Here MLFQ is an extension of MLQ scheduling where the processes can move from one queue to another queue but in MLQ scheduling processes are assigned to a fixed queue. In literature we will find by how many ways we can divide our queues, the criteria and terminology that are used till now to schedule CPU resource using the MLQ and MLFQ techniques.

A. *Basic CPU Scheduling Technique*

First come first served: In this scheduling technique the jobs in the 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 a single queue have a number of processes which are scheduled to run and all of which acquire equal and fixed 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.
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 and then the next shortest job gets the CPU time.
In this scheduling technique jobs with large burst time but higher priority face starvation problem in case of large number of small burst time processes in the queue.
Priority Scheduling: In this scheduling technique the jobs or processes are assigned a priority by the user, it can depend upon various factors such as type of process etc.

B. *Factors on Which the Efficiency of the Scheduler is Dependent*

There can be many factors which should be taken care of while working for a CPU scheduler design. Values for some of the factors should be maximized and for others their values should be decreased. Different factors includes as follows:

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 submission of requests and first response to the request.
VI. *Loss ratio:* is the fraction of requests that are dropped due to their deadline cannot be met

C. *Different Types of Process*

In multilevel CPU scheduling we divide the queue into a number of multiple queues which are divided depending upon the type of processes that are required to be scheduled. Different types of processes are as follows:

Batch Processes: Batch processes are those which do not require human interaction while performing its run, they run without human intervention.

Interactive processes: Interactive processes are those which require input from users and which interact with the system and user maximum time.

CPU Bound Processes: CPU bound processes are those processes which have maximum burst time and less required I/O or waiting time. Those processes which have these characteristics are known as CPU bound processes.

I/O Bound Processes: Processes which has maximum I/O waiting time and less CPU time required are known as I/O bound processes.

Real Time Processes: Processes which has a deadline associated with them are real time processes; they should be completed in that time required time only.

II. LITERATURE REVIEW

In proposed algorithm [1] the author has discussed about the multilevel queue management technique which is very important technique for CPU scheduling; the proposed algorithm has used a well known approach for multilevel queue scheduling is weighted fair queue (WFQ). In WFQ algorithm the queues are divided into a number of multiple queues where weights are assigned to each queue which specifies the number of jobs that are going to be scheduled from that queue in next round.

The proposed algorithm uses WFQ with fuzzy inference system and QoS measurement component.

The proposed algorithm has been named as Fuzzy Dynamic Weighted Fair Queue (FDWFQ) which places the incoming requests into different queues and weights are assigned to each queue which determines the number of jobs that are going to be scheduled next from that particular queue. The author classify incoming requests into different queues depending upon the factors deadline of processes, processing time, their ready time, and the importance level of the process, three levels are classified as three queues are being used.

The weights for different queues is determined by the author depending upon the service requirement as the author proposed this algorithm to minimize the incoming requests rejection due to their deadlines not meet which is a service requirement along with a CPU utilization factor, and the value of dynamic weights are determined by the fuzzy inference system.

Different factors used [1] by author in proposed FDWFQ algorithm for calculating weights values of different queues are:

1. IL Class: The array of structures that is used to keep QoS related parameters for each IL class.
2. DesiredLossRatio: The value that the current loss ratio desired to be less than or equal to.
3. Count_to_Total: The ratio of "count of incoming requests from this IL class" to "count of all incoming requests".
4. AvgSize_to_Total: The ratio of "average size of requests from this IL class" to "count of all incoming requests".
5. CurrentLossRatio: The ratio of "count of rejected requests from this IL class" to "count of all rejected requests".
6. Loss_Ratio_Distance (*LRD*): For each *IL* class is the deviation between average loss ratio and desired loss ratio for that class which is defined by system's administrator.
7. Busytime: The time duration that the instance is busy serving jobs.

The author has used fuzzy for designing its algorithm and has designed the Mamdani-style inference engine to calculate the dynamic weight values for different queues and the factors used are sequence number 1,3,4,6.

The membership function used here is triangular, which is required to represent the linguistic variables. The fuzzy rules are defined by the IF-Then Rule and 16 rules are defined by the author.

The author has designed an efficient algorithm in case of requests having a deadline which needs to be meet so that the loss of requests becomes less due to violation of deadline.

I In [2] the author has proposed an algorithm of MLFQ (Multilevel feedback queue) scheduling where author has divided the queue into three parts where two of three queues have Round Robin scheduling technique and the remaining one is having FCFS scheduling technique. In [2] after all the processes enter into the first queue, all the processes are sorted according to their burst time and then they are scheduled to run with a suitable time quantum.

After the processes complete their initial execution they are moved to second queue where they are further sorted according to their remaining burst if any and are scheduled for execution with a suitable time quantum value. Then the processes after their execution in second queue are moved to the third queue where they are scheduled to run with FCFS scheduling technique.

This algorithm gives less waiting and turnaround time but CPU has to wait until all the processes comes into the queue which is a limitation to this resource.

In [3] queue is divided into a number of five queues where various processes are scheduled. In [3] processes do not have any initial priority assigned to them but they are scheduled into the initial queue for the very first time they arrive, there

they are scheduled using Round Robin scheduling with a suitable time quantum value. After execution into the first queue the processes either gets completed or they are scheduled to run into the second queue. But before shifting to the next lower queue their remaining CPU burst and waiting time is calculated. In second queue they are again scheduled with Round Robin CPU scheduling technique with a suitable time quantum value and again these parameters are updated. In [3] three main parameters which are updated and calculated at each step are waiting time and remaining CPU burst time and turnaround time. After scheduling to the second queue again either processes gets completed or they are further scheduled to next lower level queues and after processes gets executed into the last queue, their remaining CPU burst is calculated again, And then every process's remaining burst time is compared with the Time quantum value of each queue and they are scheduled in that particular queue who's time quantum value suits best to that particular process's remaining burst time and then they are executed in parallel with each other; means all the queues gets independent CPU working parallel. Scheduling algorithm [3] removes the starvation problem of different processes which wait in lower level queues for very long time, by scheduling them in parallel with each other. But the number of switches required in this case is more and every time processes moves from one queue to other requires calculation and storage of different parameters as remaining burst time, waiting time and turnaround time of processes.

In [4] another algorithm with multilevel feedback CPU scheduling has been proposed where queue is divided into a number of queues and processes are moved to different queues as in [3] for execution but this [4] uses different factors which determine the priority of different processes and also the value of time quantum is variable. The proposed algorithm is for multitasking and multiprocessor environment. Initially all the processes are entered to the same initial queue where they are scheduled according to their priority which is decided by the HRRN (highest response ratio next) scheduling technique. Once processes complete their time slice in initial queue they are moved to lower level queues, the lower level queues have larger time quantum than previous queues and also priority of processes increases as they are moved to lower level queues. Once the processes are scheduled to run in initial queue the value of time quantum is calculated as per proposed algorithm in [4]; the value of ITS (intelligent time slice) is calculated depending on OTS (original time quantum), PC (Priority component), SC (shortness component), CSC (Context switch component) of each process and average the ITS of processes is assigned as the time slice for the initial queue. Further lower level queues have ITS value ($2*ITS_{prev}$) in increasing order.

Priority of processes is calculated as:

$$\text{HighestResponseRatio} = 1 + \frac{\text{waiting Time}}{\text{Expected Run Time}}$$

Response ratio of every process is calculated and process with highest value of response ratio will have highest priority and will be scheduled first for execution. Then time slice (ITS) for each queue is calculated.

OTS is calculated from some of factors as range, Number of processes, priority of current process. Where value of range is calculated as:

Range = Max burst time+ Min burst time/ Max burst time- Min burst time

The value of ITS is calculated by sum of factors as OTS, Priority component, Shortness component, Context switch component of each process which are calculated independently for each process, divided by the total number of processes. The value of ITS is then taken as the value of time quantum for the initial queue and corresponding lower level queues have time quantum value twice the value of previous queues. Here the number of queues is variable and can be decided by user; after processes gets executed in all queues but still have burst time remaining then they are scheduled to run in different queues depending on their remaining burst time value compared with the time quantum value of particular queue;

And then all queues having remaining processes will have an independent CPU for execution and there comes the multiprocessor environment same as in [3]. So this [4] multilevel feedback queue scheduling algorithm also solves the problem of starvation and at same time improves the value of turnaround and average waiting time along-with high throughput value.

III. CONCLUSION

Various algorithms for multilevel queue and multi level feedback queue scheduling are existing which improves different CPU scheduling factors as Turnaround time, waiting time, starvation problem etc. There are many techniques which are used to solve the general problem of CPU scheduling; some uses the concept of parallelism to avoid starvation while other uses the HRRN scheduling with parallelism to avoid starvation. Where other techniques provides the terminology which are best for using with QoS aware systems. After studying different issues with the algorithms discussed and the different solutions provided by different authors, there is still a scope of improvement. We can improve the algorithms by considering the vagueness of the different parameters used. In our future work we will try to propose a new fuzzy based algorithm to improve the performance of the existing system.

REFERENCES

[1] Ali Rezaee, Amir Masoud Rahmani, Sahar Adabi, Sepideh Adabi, "A Fuzzy Algorithm for adaptive multilevel Queue Management with QoS feedback", IEEE,pp.121-127,2011.

[2] Rakesh Kumar Yadav, Anurag Upadhayay, "A fresh loom for multilevel feedback queue scheduling algorithm," *International Journal of Advances in Engineering Sciences*, vol. 2, pp. 21-23, July 2012.

[3] Ayan Bhunia, "Enhancing the Performance of Feedback Scheduling," *International Journal of Computer Applications*, vol. 18, pp. 11-16, March 2011.

[4] H.S.Behera , Reena Kumari Naik, Suchilagna Parida, "A new hybridized multilevel feedback queue scheduling with intelligent time slice and its performance analysis," *Intenational Journal of engineering research and technology.*

[5] Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, *Operating System Principles,* 7th ed. , Wiley india edition, 2009.

[6] (2013) The IEEE website. [Online]. Available: http://www.ieee.org/.

[7] HRRN Scheduling (2013) [Online].Available: http://en.wikipedia.org/wiki/Highest_response_ratio_next.

[8] Weighted fair Queue (2013) [Online].Available: http://en.wikipedia.org/wiki/Weighted_fair_queuing.

