Multi-Level Queue with Priority and Time Sharing for Real Time Scheduling

Iqra Sattar, Muhammad Shahid and Nida Yasir

Abstract— **Multilevel queue scheduling and Real time scheduling is common in CPU scheduling techniques. In this paper different techniques for scheduling these algorithms has been collected and discussed. Primarily an introduction to multilevel queue and real time scheduling are discussed. In multi-level queue scheduling, the starvation problem has been solved efficiently but this technique is not suitable for real time processes. To overcome this problem, an idea of new algorithm i.e., MLQPTS (Multilevel Queue with Priority & Time Sharing Scheduling) have been proposed. In this algorithm, all the processes are listed in a queue and this queue is built depending upon the priority of each process. This priority is calculated by considering the factors such as waiting time, processing time, deadline time, etc of each process. A queue executes for a specific time called Queue Execution Time. Each process gets its time share in execution depending upon the priority-level. After each execution interval, priority of each process is re-calculated and a new queue is built which accommodates the new incoming process as well. In this way, our proposed algorithm has the properties of MLQS and it can also accommodate real time processes. The objective of the study is to have better understanding of Multi-Level Queue and Real time scheduling and to see what challenges they have to face and how these challenges are resolved by using different techniques.**

Keywords— **MLQS (Multilevel Queue Scheduling), RTS (Real Time Scheduling), MLFQS (Multilevel Feed Back Queue Scheduling) and MLQPTS (Multilevel Queue with Priority & Time Sharing Scheduling)**

I. INTRODUCTION

One of the constant challenges for multi-level queue scheduling is to minimize resource starvation and to ensure fairness amongst the parties utilizing the resources and for real time systems is to build a platform that can meet timeliness requirement of system. After having a look on these two scheduling algorithms we also introduce a new scheduling technique that will overcome the problems of real time scheduling and multi-level queue scheduling algorithms to some extent. For Multi-programmed operating system CPU scheduling is the basic requirement. To obtain the maximum CPU utilization CPU is switched among various processes, in this way system become more productive. Scheduling is a policy that guarantees that no job waits indefinitely for a service. CPU is one of most important resources which require scheduling, on which the working and speed of system depends. Scheduling deals with the problem of deciding which of the outstanding requests is to be allocated resources. In multi-programmed system when the CPU becomes idle operating system select one of the processes that are in the ready queue to be executed. CPU scheduling is important because it can have a huge effect on resource utilization and on overall performance of the system [8]. There are three different types of scheduler long term (job scheduler), medium scheduler and short term (job scheduler). Short term scheduler selects the processes from the ready queue and then it's the duty of dispatcher to allocate CPU to selected process. Different CPU scheduling algorithm exists for different environments. The criteria used for scheduling algorithm optimization include maximum CPU utilization, maximum throughput, smallest turnaround period, least waiting time, and minimum reply time. There is no universal best scheduling algorithm and many operating systems use extended or combinations of the scheduling algorithms.

Scheduling of CPU resource has many ways by which it can be scheduled like FIFO, Round Robin, Shortest Job First Priority queue and so on, But scheduling a CPU which has different type of processes, which are required to run can be scheduled using Multi level queue scheduling.

The Multi-level Queue scheduling is considered to be superior due to its better management of variety of processes. Multi-level Queue scheduling is intended to meet the following design requirements for multimode systems: Give preference to small jobs and I/O bound processes. Processes are divided into categories based on their need for the CPU. Other scheduling algorithm is real time scheduling. A real-time system is a system that is required to complete the task within time intervals directed by the environment [1].

Real-time systems are those whose correctness depends not only on logical results of computations, but also on the time at which the results are produced.

A. First we have a look on basic CPU scheduling techniques

First Come First Serve: It is a traditional scheduling technique in which all the jobs have same priority, job queue

Iqra Sattar is with the Department of Computer Science & Engineering, University of Lahore (Sargodha Campus), Pakistan (Email: iqrasatar@gmail.com)

Muhammad Shahid is with Department of Electrical Engineering and Computer Science, Pakistan Institute of Engineering and Applied Sciences, Pakistan (Email: shahidbhutta@gmail.com)

Nida Yasir is with the Department of Computer Science & Engineering, University of Lahore (Sargodha Campus), Pakistan (Email: nida.yasir@yahoo.com)

scheduled the jobs in the order in which they come, job which come first will take CPU time first then next and so on, processes are inserted into the tail of a queue when they are submitted.[9] The next process is for execution is taken from the head of the queue.

Round Robin Scheduling: In this all the processes in the job queue gets an equal amount of CPU time. After the time expires, the process is preempted and added to end of ready queue. The scheduler goes around this queue, allocating the CPU to each process for a time interval of allocated quantum. New processes are added to end of the queue [10].

Shortest Job First: The process which has short burst time will schedule first to run then next shortest job gets the CPU time. A scheduler arranges the processes with the least burst time in head of the queue and longest burst time in tail of the queue. This requires advanced knowledge or estimations about the time required for a process to complete [9]. This algorithm is designed for maximum throughput in most scenarios.

Priority Scheduling: In this scheduling jobs are assigned the priority by the user. The OS assigns a fixed priority rank to each process. Lower priority processes get interrupted by incoming higher priority processes.

B. Multi-Level Queue Scheduling

The situation in which the process is divided into different groups, multi-level queue scheduling is used. The characteristics of multi-level scheduling are as follows (Fig. 1):

Based on the types, the processes are divided into different queue. Processes are assigned to one queue permanently. The scheduling algorithm is each queue is unique. For example, as shown in the figure below, interactive process and batch job may use round robin scheduling method and FCFS method respectively.

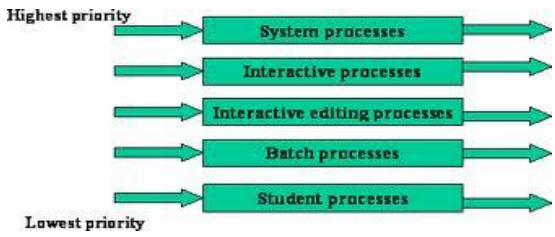

Fig. 1: Multi-level queue scheduling

Also, the queue must be scheduled and generally, it has fixed priority. A common division between foreground processes and background processes is made. These two types of processes may differ in their response times; therefore, they may need different scheduling. The foreground processes may have priority over background processes and this priority may be defined externally. The batch queue can only be executed when the queue for system executes and the interactive processes are idle or empty. If the batch process is still in process and the interactive process enters the ready queue, the batch will be prevented.

C. Real Time Scheduling

A real Time System is a system in which the timing constraints are strict. The output results of these systems depend upon the algorithm of computation and the time duration utilized to generate certain results. If the timing constraints are violated, the obtained results are no longer valid. Therefore, it is necessary to abide by the timing condition. Also, the algorithm must efficient enough to fully utilize the available resources [2] – [6]. Each process has timing properties. These properties must consider while scheduling and executing on real time system. These properties include: release time / ready time, deadline, execution time, completion time, finishing time, penalty factor etc [4] – [7].

A real-time system will usually have to meet many demands within a limited time. Thus, the allocation of the system resources needs to be planned so that all demands are met by the time of their respective deadlines. This is usually done using a scheduler which implements a scheduling policy that determines how the resources of the system are allocated to the demands.

RTS can be divided further into two categories: There are two types of Real Time Systems: (i) Hard Real Time Systems and (ii) Soft Real Time Systems. Hard Real Time Systems are those in which the deadline condition must be fulfilled or else there will be undesirable consequences such fatal error or damage to the system. Soft Real Time Systems are those in which the deadline condition can be compromised up to an allowable limit [4], [5], [6].

Analysis for real-time scheduling: There are a number of algorithms available that loads the CPU up to 100% to increase the utilization factor. The processes are scheduled with respect to Earliest Deadline First (EDF) and Least Laxity First (LLF). These both algorithms are optimal. EDF algorithm searches through all the process and executes the process which the earliest deadline. LLF algorithm schedules and executes the process which has least laxity.

II. LITERATURE SURVEY

In this section different methods proposed by different authors for scheduling these two techniques has been collected and discussed.

A. MLFQ (Multi Level Feed Back Queue) Scheduling Algorithm

MLFQ using Three queues: In MLFQ (Multi-level feedback queue) scheduling, the queue is divided into three parts where two queues have Round Robin scheduling technique and the remaining one has FCFS scheduling technique [15]. All the processes are sorted in first queue according to their burst time and then they are allowed to execute for a specific time. When the processes complete their

initial execution, they are sorted in second queue according to their remaining burst time. After the execution, the processes are moved to the third queue where they are sorted to run with FCFS scheduling technique. This algorithm helps to minimize the waiting time and turnaround time but CPU has to wait to build a queue of all the processes. It is the main cause that limits the best utilization of resources.

MLFQ using Five queues: In this technique, the processes are scheduled in five different queues. Initial priority is not assigned to the process but they are scheduled using Round Robin scheduling with a suitable time quantum value at the time when the process is initiated [16]. The processes are scheduled and allowed to run in a queue according to their burst time. In the end, either the processes are completed or they are sorted to run into the second queue to run again with the remaining burst time and waiting time. This scheduling is done using Round Robin CPU scheduling technique which also assigns a suitable time quantum value. Waiting time and remaining CPU burst time and turnaround time are the main parameters that are calculated and updated in each step. At the end of second queue, either the processes get completed or they are further sorted to next queue till all the processes get executed. Some processes have to for long time for execution. This scheduling algorithm executes all the processes in parallel. In this way, this algorithm removes the starvation problem of different processes. In this case the number of switches is more because of the storage and calculations of burst time and waiting time of each queue.

B. WFQ (Weighted Fair Queue) Scheduling Algorithm

In the proposed algorithm [14], the multi-level queue management technique is described. It is very critical technique for CPU process scheduling. In the proposed algorithm, weighted fair queue (WFQ) is used for multilevel queue scheduling. WFQ algorithm divides the queue into a number of multiple queues and each queue is assigned a weight that defines the number of jobs to be scheduled in a queue for next round. WFQ is used with fuzzy inference system therefore, the newly proposed algorithm is named as Fuzzy Dynamic Weighted Fair Queue (FDWFQ) that schedules the incoming requests into different queues and assigns a weight to each queue that determines the number of jobs to be scheduled next from that particular queue. The incoming requests are arranged into different queues depending upon the nature of the process. The weights for different queues are determined accordingly. This algorithm tends to minimize the incoming requests rejection the fuzzy model is used to design this algorithm. Mamdani-style inference engine is employed to calculate the dynamic weight for each queue. The design is efficient for request management to meet the deadline and in this way the loss of requests is minimized.

C. Earliest-Deadline-First Scheduling (EDF)

Liu and Leyland proposed EDF Scheduling [11]. EDF algorithm searches through all the process and executes the process which the earliest deadline. The algorithm is as follows:

Step 1: Load all the tasks and determine their characteristics such as start time, end time, remaining time and deadline.

Step 2: Check if the system is in idle state, then start task processing and after processing go to step 4. If there is no process to process then go to step 3.

Step 3: If a new task with earliest deadline enters, then update the remaining time of the process that is in process and exchange it with the new process. Else update the starting time of new process.

Step 4: Go to step 2 if the tasks are not scheduled. Else stop.

This algorithm fully utilizes the available resources of CPU and all the deadlines of the process can be met. This is the main advantage of this algorithm

D. Least Laxity First (LLF)

This algorithm is proposed by David B. Stewart and Pradeep K. Khosla [12]. In this algorithm, the laxity to each task is assigned and the task with minimum laxity is processed first. This is the reason that this algorithm is call Least Laxity First algorithm. The laxity of a task is defined by the difference of deadline time and remaining time of computation. The priority is assigned according to the laxity of each process [13]. There is a phenomenon that if a process loads and it has smaller laxity as compared the task which is in process then the task with greater laxity will be closed and its remaining time will be assigned after calculation and the task with smaller laxity will start execution to meet the deadline. It is an ideal algorithm for the system which processes the periodic real time tasks.

III. PROPOSED ALGORITHM

The starvation problem has been solved efficiently by MLQS but it is not suitable for real time processes. To accommodate both of them, MLQPTS (Multilevel Queue with Priority & Time Sharing Scheduling) have developed in which all the tasks are scheduled in a queue depending upon the priority level. This priority level is defined from the characteristics of the process. The process with greater priority will get major share in execution time to meet the deadline condition. In this way, this algorithm will meet the deadline condition which is the property of real time systems. Hence the real time task can also be executed using multi-level queue technique (Fig. 2):

Step 1: Create a list of all the tasks. If a new task enters, it is also included in the list.

Step 2: Calculated the priority index for each task depending upon the properties of tasks such as deadline time, waiting time, response time etc.

Step 3: Build a queue depending upon the priority of tasks. A queue runs for a specific time. Each task gets its execution time share depending upon the priority level. After that, go to Step 2.

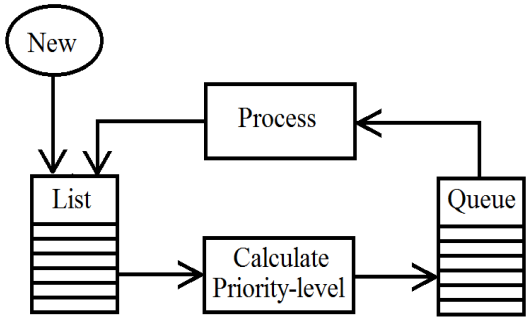

Fig. 2: Proposed Algorithm (MLQPTS)

In this algorithm, initially all the tasks enter into the list. From the list priority level of all the tasks that exists in the list is calculated. The priority level depends upon the characteristic of the processes such as their waiting time, turnaround time, deadline time etc. After that a queue is built depending upon the calculated priority level and each queue executes for a specific time called Queue Execution Time. Each task gets its time share in execution but the task having the highest priority will execute first having major time in execution to meet the deadline condition. After each execution interval, priority level of each process is re-calculated, the processes that need to be re-scheduled is again listed and a new queue is built which accommodates the new incoming process as well and the whole process starts again.

Key Features:

- It removes the starvation problem. All the tasks get their time share in queue execution interval.
- Critical tasks which has earliest deadline are executed first to meet the deadline, therefore, this algorithm is suitable for real time tasks.
- It builds a new queue after each queue execution interval; the CPU utilization factor might be slightly low due to re-calculation of priority level again and again till completion of processes.

IV. CONCLUSION & FUTURE WORK

Different algorithms exists for Real time and Multilevel queue scheduling which improves different factors like waiting, turnaround time and starvation etc but there is a still scope of improvement. Multilevel queue scheduling removes the starvation problem of different processes but this technique is not suitable for real time processes. To overcome this problem, we have proposed Multilevel Queue with Priority & Time Sharing for Real time scheduling which has the properties of MLQS and also accommodate real time processes.

In our future work we will try to improve and carry on our idea in detail to increase the performance of the existing system and to have maximum CPU utilization. We will also implement our design on physical hardware or in some real environment and try to remove the limitations that will come during this process.

REFERENCES

[1]. A. Burns and A. J. Wellings. Real Time Systems and Programming Languages Addison-Wesley, England, 2001.

[2]. J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, “Real-Time:Theory in Practice,”Preceedings of REX Workshop, Mook, The Netherlands, Springer-Verlag company, June 3-7, 1991.

[3]. G. C. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications,” Springer, September, 2006.

[4]. M. Joseph, “Real-time Systems: Specification, Verification and Analysis,” Prentice Hall, 1996.

[5]. P. A. Laplante, “Real-time Systems Design and Analysis, An Engineer Hand-book,”IEEE Computer Society, IEEE Press, 1993.

[6]. J. A. Stankovic and K. Ramamritham, “Tutorial on Hard Real-Time Systems,”IEEE Computer Society Press, 1988.

[7]. W. Fornaciari and P. di Milano, “Real Time Operating Systems Scheduling Lecturer,” www.elet elet.polimi polimi.it/ fornacia it/ fornacia.

[8]. Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F. Safaei, (2005). Processing resource scheduling in programmable networks. Computer communication, 28:676-687

[9]. Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India,

[10]. Sun Huajin’, Gao Deyuan, Zhang Shengbing, Wang Danghui; “ Design fast Round Robin Scheduler in FPGA”, 0-7803-7547-5/021/$17.00@2002 IEEE

[11]. Yoo, Myungryun and M. Gen, “ Study on Scheduling for Real- time Task by Hybrid Multiobjective Genetic Algorithm”, Thesis, 2006

[12]. David B. Stewart, Pradeep Khosla, “Real -Time Scheduling of Sensor-Based Control Systems”, 1991

[13]. Arezou Mohammadi and Selim G. Akl, “Scheduling Algorithms for Real-Time Systems”, Technical Report No. 2005-499, July 15, 2005

[14]. 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.

[15]. 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.

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