Studying Main Differences between Multilevel Queue (MLQ) and Multilevel Feedback Queue (MLFQ)

K. Kala Bharathi

Department of Computer Science, St. Pious X Degree & P. G. College for Women, Nacharam, Hyderbad-500 076, India

kala_bharathi31@yahoo.com

Abstract- **Comparisons between the Multilevel Queue and Multilevel Feedback Queue CPU scheduling algorithm. It is a long time running discussion in scheduling algorithms to decide which of the processes in the ready queue is to be allocated the CPU first. However there exist some problems with these algorithms when facing the fast growth of real-time systems and handhelds, in which requirements for interactivity and the growth of system loads need to be taken into corresponding consideration and in my approach I proposed which is best.**

Keywords- **CPU scheduling, Quantum, Burst time**

I. INTRODUCTION

When a computer is multi programmed, it frequently has multiple processes competing for the CPU at the same time. This situation occurs whenever two or more processes are simultaneously in the ready state. If only one CPU is available, a choice has to be made which process to run next. The part of the operating system that makes the choice is called the scheduler and the algorithm it uses is called the scheduling algorithm [1].

In a single-processor system, only one process can run at a time, any others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization [2]. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best in CPU Utilization, Throughput, Turnaround time and Response time [2].

A. CPU utilization

We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 %. In a real system, it should range from 40 % (for a lightly loaded system) to 90 % (for a heavily used system).

B. Throughput:

If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called *throughput.*

C. Turnaround time:

The interval from the time of submission of a process to the time of completion is the *turnaround time.* Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

D. Waiting time:

Waiting time is the sum of the periods spent waiting in the ready queue.

E. Response time:

In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called *response time,* is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device [2]. It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. In most cases, we optimize the average measure. However, under some circumstances, it is desirable to optimize the minimum or maximum values rather than the average. For example, to guarantee that all users get good service, we may want to minimize the maximum response time [2].

Among these entire things Multilevel queue algorithm (MLQ) and Multilevel Feedback queue (MLFQ) differ in Level of priority, time quantum and allocating of jobs. Comparisons of these two scheduling algorithms tend to reflect their origins and their representation.

II. Essential Differences Between Multilevel Queue (MLQ) And Multilevel Feedback Queue (MLFQ)

1.In Multilevel queue (MLQ) processes are classified into different groups. For example, common division is made between foreground (interactive) processes and background (batch) processes which have different response time and scheduling needs. In addition

In Multilevel Feedback queue (MLFQ) it allows a process to move between the queues, according to the characteristics of their CPU burst.

3. In Multilevel queue (MLQ) the foreground queue might be scheduled by Round Robin algorithm while the back ground queue is scheduled by First Come First Serve algorithm. There is possibility of starvation.

But in Multilevel Feedback queue (MLFQ) if a process uses too much CPU time it will be moved to a lower-priority queue. This schema leaves I/O bound and interactive processes in the higher priority queues. In addition, a process that waits too long in a lower priority queue may be moved to a higher-priority queue preventing starvation.

III. Examples

Multilevel queue (MLQ) algorithm with five queues, listed below with order of priority:

a) System processes
b) Interactive processes
c) Interactive editing processes
d) Batch processes
e) Student processes

Algorithm chooses the process from the occupied queue that has the highest priority, and run that process either Preemptive or Non-preemptively

Each queue has its own scheduling algorithm or policy.

Possibility-I

Each queue has absolute priority over lower-priority queues then no process in the queue could run unless the queues for the highest-priority processes were all empty.

For example, in the below Fig. 1 no process in the batch queue could run unless the queues for system processes, interactive processes and interactive editing processes will all empty.

Possibility-II

If there is a time slice between the queues then each queue gets a certain amount of CPU times, which it can then schedule among the processes in its queue. For instance;

foreground processes may have priority over background [2].

But, in Multilevel Feedback queue (MLFQ), it contains two queues, lower-priority queues and higher-priority queues. In this the separation of processes are done according to the characteristics of their CPU bursts.

2. In Multilevel queue (MLQ) the processes are permanently assigned to one queue based on their memory size, process priority or process type.

- 80% of the CPU time to fore ground queue using Round Robin (RR).
- 20% of the CPU time to back ground queue using First Come First Serve (FCFS).

Since processes do not move between queues so, this policy has the advantage of low scheduling overhead, but it is inflexible.

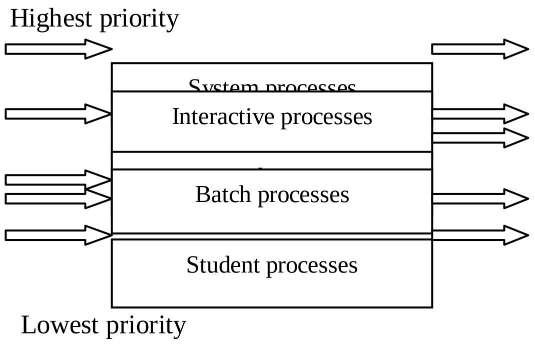

Fig. 1: Multilevel queue scheduling

No process in the batch queue could run unless the queue for system processes and interactive processes were all empty. If an interactive process enters the ready queue while a batch process was running, the batch would be preempted

Now we will see the example to explain multilevel feedback queue (MLFQ). It contains three queues numbered from 0 to 2.

- ➢ Three queues:
- *Q*0 - Round Robin (RR) with time quantum 8 milliseconds
- *Q*1 - Round Robin (RR) time quantum 16 milliseconds
- *Q*2 - First Come First Serve (FCFS)
- ➢ Scheduling
- A new job enters queue *Q0* which is served *Q2*. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue *Q*1.

- At *Q*1 job is again served *Q2* and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue.

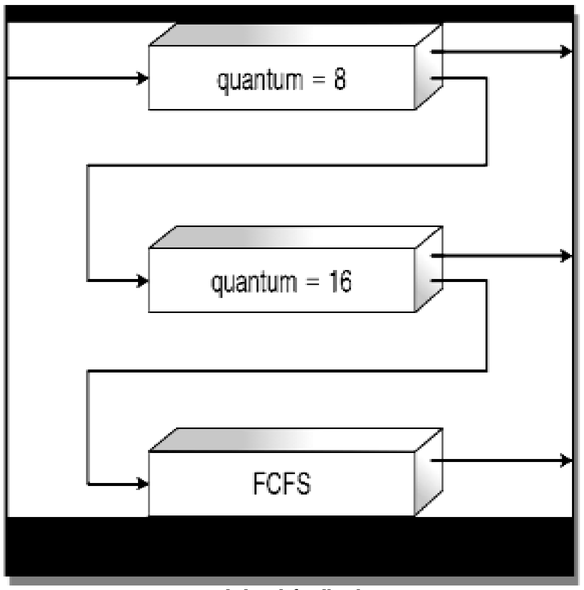

Fig. 2: Multilevel feedback queues

IV. CONCLUSION

Multilevel Feedback Queue (MLFQ) is interesting because instead of demanding *a priori* knowledge of the nature of a job, it instead observes the execution of a job and prioritizes it accordingly. In this way, it manages to achieve the best of both worlds, it can deliver excellent overall performance (similar to SJF/STCF) for short-running interactive jobs, and is fair and makes progress for long-running CPU-intensive workloads. For this reason, many systems, including BSD UNIX derivatives [LM+89, B86], Solaris [M06] and Windows NT and subsequent Windows operating systems [CS97] use a form of MLFQ as their base scheduler.

ACKNOWLEDGMENT

I thankful to The Principal, St. Pious X Degree & P.G. College for Women for providing literature facilities and also I thankful to my colleagues for encouraging me in this work.

REFERENCES

[1] S. Andrew Tanenbaum, *Modern Operating Systems*, 2nd ed., Prentice-Hall of India Private Limited, New Delhi

[2] Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, *Operating System Concepts*, 7th ed., John Wiley & Sons (Asia) Pvt. Ltd., Singapore.