A 2LFQ Scheduling with Dynamic Time Quantum using Mean Average

Rakesh K. Lenka
Student
CSED, MNNIT
Allahabad, India

Prabhat Ranjan
System Manager
CSED, MNNIT
Allahabad, India

ABSTRACT
The efficiency and performance of multitasking operating systems essentially depends on the nature of CPU scheduling algorithm. There are many algorithms available for CPU scheduling. Each having its own deficiency and limitations. One of the most well-known approaches for scheduling is the Multi-level Feedback Queue (MLFQ). The MLFQ tries to work in a two-fold manner. First, it tries to optimize turnaround time as it is done by running shorter jobs first. Unfortunately, the OS doesn't generally have the knowledge that how long a job will run for, exactly the knowledge that algorithms like SJF (or SRTF) require. Second, MLFQ attempts to make a system feel responsive to interactive users (i.e. users sitting and staring at the screen, waiting for a process to finish), and thus minimize response time. Well-known algorithms like Round Robin also reduce response time but are less suitable for turnaround time. In this paper, we proposed a new approach for feedback scheduling algorithm which helps to improve the efficiency of CPU. The paper presents an approach called dynamic-time-quantum 2LFQ (Two-level Feedback Queue) scheduling. The idea is to make the operating systems adjusts the time quantum according to the burst time of set of waiting processes in the ready queue.

Keywords
CPU scheduling, dynamic-time-quantum, scheduling algorithm.

1. INTRODUCTION
A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. Another main reason is the need for processes to perform I/O operations in the normal course of computation. Since I/O operations ordinarily require orders of magnitude more time to complete than do CPU instructions, multiprogramming systems allocate the CPU to another process whenever a process invokes an I/O operation.

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 [1]. 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. Some of the criteria include Utilization/Efficiency: keep the CPU busy 100% of the time with useful work, Throughput: maximize the number of jobs processed per hour, Turnaround time: from the time of submission to the time of completion - minimize the time batch users must wait for output, Waiting time: Sum of times spent in ready queue - minimize this, Response Time: time from submission till the first response is produced - minimize response time for interactive users, Fairness: make sure that each process gets a fair share of the CPU.

There exist a different scheduling algorithms, each one of these algorithms has advantages and disadvantages and as follows: First-Come-First-Served (FCFS) : it is easy to implement, but it ignores the service time request and all other criteria that may influence the performance with respect to turnaround or waiting time which is not suitable in real time applications. This is mainly because one process can monopolize CPU with long execution time that may hinder many short processes to complete before deadline.

On the other hand, priority scheduling: allocates processes to the CPU on the basis of an externally assigned priority and run the highest-priority first. The key to the performance of priority scheduling is in choosing priorities for the processes. But it cause low-priority processes to starve and the solution of this problem is aging operation. Another problem with priority scheduling is deciding which process gets which priority level assigned to it.

Shortest-Job-First (SJF): scheduling is giving the optimal, providing the shortest average WT. The obvious problem with this algorithm is that it is require precise knowledge of how long a job or process will run and this information is not usually available and unpredictable. The shortest remaining time first (SRTF) scheduling algorithm is a preemptive version to an older non-preemptive algorithm known as shortest job first scheduling. Shortest job first scheduling runs a process to completion before running the next one. The queue of jobs is sorted by estimated job length, so that short programs get to run first and not be held up by long ones. This

minimizes average response time. The disadvantages of SRTF scheduling algorithm is that long-burst (CPU-intensive) processes are hurt with a long mean waiting time. In fact, if short-burst processes are always available to run, the long-burst ones may never get scheduled. Moreover, the effectiveness of meeting the scheduling criteria relies on our ability to estimate the CPU burst time.

Round robin (RR) scheduling is a preemptive version of first-come, first-served scheduling. Processes are dispatched in a first-in-first-out sequence but each process is allowed to run for only a limited amount of time. This time interval is known as a time-slice or quantum. If a process does not complete or get blocked because of an I/O operation within the time slice, the time slice expires and the process is preempted. Process gets blocked because of an I/O operation, it is then preempted. This preempted process is placed at the back of the ready list where it must wait for the processes that were already on the list to cycle through the CPU. Round robin scheduling is fair in that every process gets an equal share of the CPU. It is easy to implement and, if we know the number of processes on the ready list, we can know the worst-case response time for a process. The disadvantages of RR scheduling algorithm is Giving every process an equal share of the CPU is not always a good idea. For instance, highly interactive processes will get scheduled no more frequently than CPU-bound processes.

In Multilevel Feedback Queue (MLFQ): processes are scheduled according to their remaining CPU burst and they are shifted down from queue to queue as they have some remaining CPU burst. Every queue has unique time slice that gradually increases from upper level queue to lower level queue. So the CPU intensive jobs go down from upper queues to lower queues gradually for getting completed. Thus, lower priority queues are filled with CPU intensive jobs and as a result these processes start to starve for getting CPU attention. So then it will follow first come first serve scheduling among these jobs. It can deliver excellent overall performance similar to SJF or SRTF scheduling for turnaround time, while it can also provide a responsive system for interactive jobs just like Round Robin scheduling. Here interactive job means the jobs which go for input and output operations frequently compare to the jobs which are more focused on getting CPU cycles which are considered as CPU intensive jobs. For this reason, many systems, including BSD Unix derivatives, Solaris, and Windows NT and subsequent versions use a form of MLFQ as their base scheduler. Multi-level feedback queues are good for separating processes into categories based on their need for a CPU. They favor I/O bound processes by letting them run often. Versions of this scheduling policy that increase the quantum at lower priority levels also favors CPU bound processes by giving them a larger chunk of CPU time when they are allowed to run. The obvious problem with this algorithm is that the priority scheme here is one that is controlled by the system rather than by the administrator or users. A process is deemed important not because it is, but because it happens to do a lot of I/O. This scheduler also has the drawback that I/O bound processes that become CPU bound or CPU bound processes that become I/O bound will not get scheduled well [2].

In this paper, a new algorithm is proposed to solve the constant time quantum problem. The algorithm is based on dynamic time quantum approach where the system adjusts the time quantum according to the burst time of the existed set of processes in the ready queue. The section 2 states the related works done in this field. Section 3 describes the proposed method in details. Section 4 discusses the simulation done in this method, before concluding this paper in the last section.

2. RELATED WORKS

In past few years different approaches are proposed to increase the performance of MLFQ scheduling in different ways. Rami J. Matarneh [4], proposed a method to assign a dynamic time quantum for the RR algorithm instead of fixed time quantum, where the operating system itself can finds the optimal time quantum without user intervention. Ajit. Singh, proposed an approach for RR scheduling algorithm, which helped to improve the CPU efficiency in real time and time-sharing operating systems, the authors mentioned that their results were strange through different experimental cases [5].

Rami J. Matarneh [6] founded that an optimal time quantum could be calculated by the median of burst times for the set of processes in the ready queue, unless if this median is less than 25ms. In such case, the quantum value must be modified to 25ms to avoid the overhead of context switch time [6].

In paper [7], Recurrent Neural Network has been used to optimize the number of queues and quantum of each queue of MLFQ scheduler to decrease response time of processes and increase the performance of scheduling. In this paper the proposed neural network takes inputs of the quantum of queues and average response time. After getting the required inputs, it takes the responsibility of finding relation between the specified quantum changes with an average response time. It can find the quantum of a specific queue with the help of optimized quantum of lower queues. Thus, this network fixed changes and specify new quantum, which overall optimize the scheduling time.

Rakesh Mohanty [3] used the median approach and have obtained good results. On the other hand, Helmy [12] proposed a weighting technique for RR algorithm, as an attempt to make a combination between the low scheduling overhead of RR algorithms and favor short jobs. Higher process weights means relatively higher quantum of time and the small processes will be given more time, so that they will be removed earlier from the RQ.

In Basney [10], smoothed competitive analysis is applied to multilevel feedback algorithm. Smoothed analysis is basically mixture of average case and worst case analysis to explain the success of algorithms. This paper analyses the performance of multilevel feedback scheduling in terms of the time complexity. Any performance enhancing approach can use

this approach for performance analysis in terms of the time complexity.

Mohanty and others also developed other algorithms in order to improve the scheduling algorithms performance [8], [9] and [11]. One of them is constructed as a combination of priority algorithm and RR [8] while the other algorithm is much similar to a combination between SJF and RR [9].

3. PROPOSED ALGORITHM

The proposed scheduling algorithm is based on dynamic time quantum. The proposed architecture consists of 2 Queues as shown in figure 1.This paper present a solution to the time quantum problem by making the operating system adjusts the time quantum according to the burst time of the set of processes existed in the Queue-1. When the operating system is installed for the first time, it begins with time quantum equals to the burst time of the first dispatched process. The next time quantum is determined dynamically after the end of first time quantum. Repeatedly when a new process is loaded into Queue-1 in order to be executed, the operating system calculates the average of sum of the burst time of processes found in ready queue including the newly arrived process. If the CPU burst time of the process exceeds the dynamic time quantum, processor will switch from that process and start executing the next process in the Queue-1.The preempted process added to Queue-2 in ascending order of their remaining burst time. The processes in the Queue-2 can be executed if there is no process left in Queue-1.

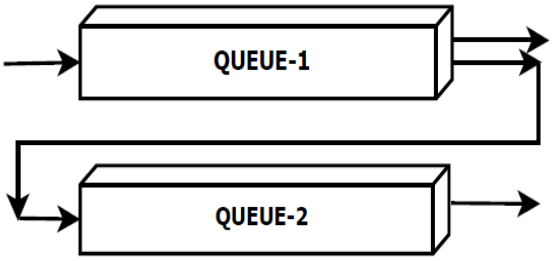

Fig. 1: Proposed Scheduling Model

3.1 Proposed Algorithm Pseudo Code

Algorithm 1: Pseudo code of the Proposed algorithm.

```
1.  New process Pi arrives
2.  Pi enters to Q-1 in ascending order of BT[i]
3.  Calculation of dynamic time quantum (DTQ)
4.  Assign the dynamic time quantum (DTQ) to all process
5.  While (Queue-1 != Empty)
6.      If ( BT[i] <= DTQ )
7.          Allocate Pi to CPU till completion
8.      Else
9.          The process will occupy the CPU till DTQ
10.         Set BT[j]= DTQ-BT[j]
11.         Pj added to Queue-2 in the ascending order of BT[j]
12. While (Queue-1= Empty  && Queue-2 != Empty)
13.     Assign the process Pk  to CPU till completion
14. Calculate RT, WT, TAT, CS
```

3.2 Proposed Algorithm Flowchart

The flowchart of proposed scheduling algorithm is shown in figure 2.

4. SIMULATION RESULT

The proposed 2LFQ algorithm is implemented using C++. For evaluation of the proposed approach two different cases with random burst are considered. In first case we took a group of five processes with burst time 20, 5, 8, 7, 14 are taken.

The obtained results are compared with the traditional approaches like First-Come-First-Served, Shortest-Job-First, Round Robin with time quantum 10 units. The comparison of waiting time of the proposed algorithm with the existing algorithm is shown in Table 1 and Figure 3. The comparison of turnaround time of the proposed algorithm with the existing algorithm is shown in Table 2 and Figure 3.

As a second case a group of four processes with burst time 20, 40, 60, 80 are randomly taken. The comparison of waiting time, turnaround time, no of context switch of the proposed algorithm with other research's result presented in [4] and [5] are shown in Table 3 and Figure 4. It is clearly observed from the Table 1, Table 2 and Figure 3, the turnaround time, waiting time and response time of the processes are optimum for the proposed algorithm compared to all the other fundamental algorithms. It is also clearly observed from the Table 3 and Figure 4, the turnaround time, waiting time and the number of context switch of the proposed algorithm are minimum when compared with the results presented in [4] and [5].

Table 1. WT for individual process and average WT for each scheduling method

PROCESS ID	WAITING TIME				
	BT	FCFS	SJF	RR	PROPOSED
1	25	0	34	44	34
2	5	25	0	10	12
3	8	30	12	15	17
4	7	38	5	23	25
5	14	45	20	40	32
AVG	11.8	27.6	14.5	26.4	24

Table 2. TAT for individual process and average TAT for each scheduling method

PROCESS ID	TAT				
	BT	FCFS	SJF	RR	PROPOSED
1	25	25	59	69	59
2	5	30	5	15	17
3	8	38	20	23	25
4	7	45	12	30	32
5	14	59	34	54	46
AVG	11.8	39.4	26	38.2	35.8

Table 3. Results Comparison

	RR	METHOD[4]	METHOD[5]	PROPOSED
TAT	120	112.5	114	112.5
WT	70	77.5	83	62.5
CS	9	6	7	5

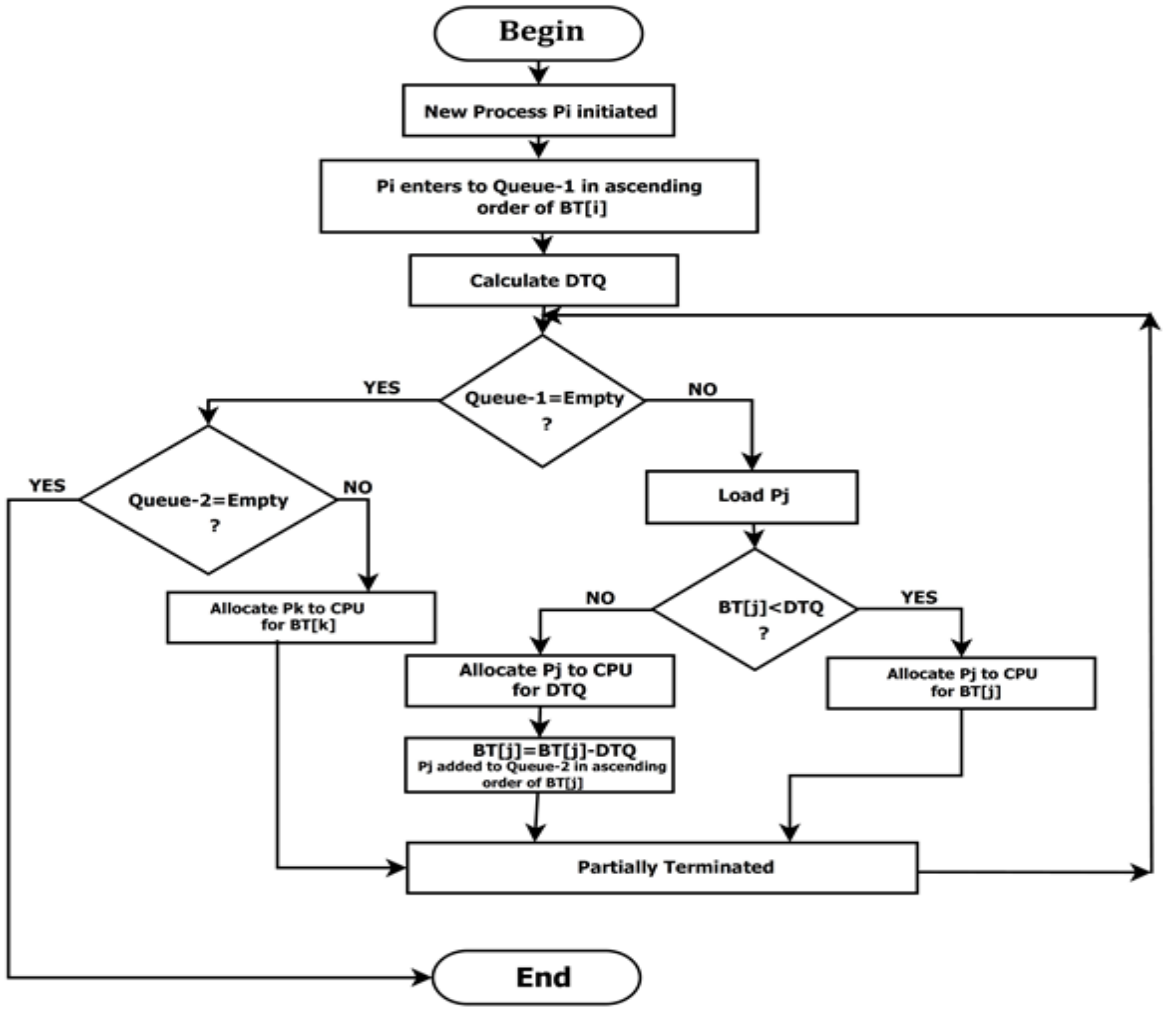

Fig. 2: Proposed Scheduling Flowchart

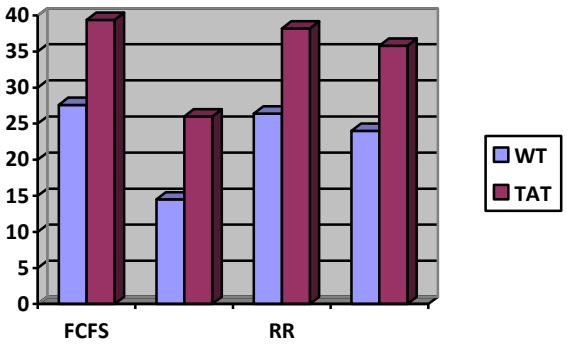

Fig. 3: Results comparison of case 1

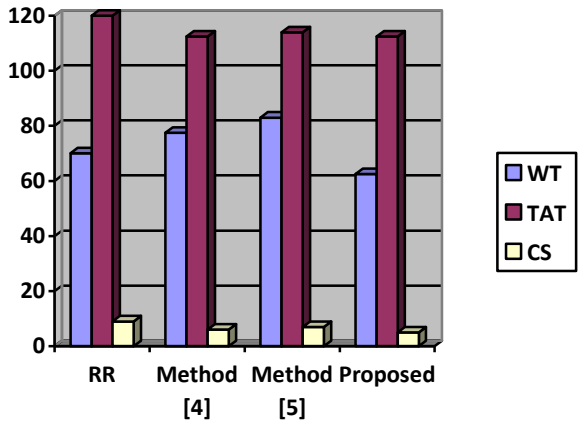

Fig. 4: Results comparison of case 2

5. CONCLUSION

Since selection of optimal time quantum is an important issue in most of the scheduling algorithms that are based on Round Robin technique, our approach attempts to answer this important issue by using dynamic time quantum instead of fixed time quantum, where the operating system itself finds the time quantum without user intervention. The approach 2LFQ extends the performance of feedback scheduling algorithm by minimizing the response time and overall turnaround time of the system. It may concluded from the simulation study come so far that the number of context switch is also minimum compared to other approaches.

We are now working on the behavior of the 2LFQ scheduling by varying the number of process with random burst. Hence in future the proposed algorithm will be implemented and can be tested in open environment.

6. REFERENCES

[1] Silberschatz, A., P.B. Galvin and G. Gagne, 2004, "Operating Systems Concepts", 7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663, pp. 944.

[2] Tanebaun, A.S., 2008, "Modern Operating Systems", 3rd Edn., Prentice Hall, ISBN: 13: 9780136006633, pp. 1104.

[3] Rakesh Mohanty, H. S. Behera, Debashree Nayak,"A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 5– No.5, 2010.

[4] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences 6 (10): 1831-1837, 2009.

[5] Ajit. Singh, P. Goyal, S. Batra, "An Optimized Round Robin Scheduling Algorithm, for CPU Scheduling", IJCSE, Vol. 02, No. 07, 2010.

[6] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences, Vol 6, No. 10, 2009.

[7] Becchetti, L., Leonardi, S. and Marchetti S.A. (2006), "Average-Case and Smoothed Competitive Analysis of the Multilevel Feedback Algorithm" Mathematics of Operation Research Vol. 31, No. 1, February, pp. 85–108.

[8] Rakesh Mohanty, H. S. Beheram Khusbu Patwarim Monisha Dash, M. Lakshmi Prasanna , "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011.

[9] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", In Proceedings of International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010.

[10] Basney, Jim and Livny, Miron (2000), "Managing Network Resources in Condor", 9th IEEE Proceedings of the International Symposium on High Performance Distributed Computing, Washington, DC, USA.

[11] Rakesh Mohanty, H. S. Behera, Debashree Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 5– No.5, August 2010.

[12] Helmy, T. and A. Dekdouk, Burst round robin as a proportional-share scheduling algorithm. IEEEGCC, King Fahed University, 2007. http://eprints.kfupm.edu.sa/1462.