An Optimum Multilevel CPU Scheduling Algorithm

Sindhu M
Department of Information Technology,
Indira Institute of Engineering & Technology,
Thiruvallur, TN, India,
sindhumoha@gmail.com

Rajkamal R
Department of Electrical and Electronics Engineering,
Indira Institute of Engineering & Technology,
Thiruvallur, TN, India,
rajkamalr@gmail.com

Vigneshwaran P
Department of Computer Science and Engineering,
Indira Institute of Engineering & Technology,
Thiruvallur, TN, India,
vigneshwaran05@gmail.com

***Abstract*- Central Processing Unit *(*CPU) scheduling plays a deep-seated role by switching the CPU among various processes. As processor is the important resource, CPU scheduling becomes very important in accomplishing the operating system (OS) design goals. The intention of the OS should allow the process as many as possible running at all the time in order to make best use of CPU. The high efficient CPU scheduler depends on design of the high quality scheduling algorithms which suits the scheduling goals. In this paper, we proposed an algorithm which can handle all types of process with optimum scheduling criteria.**

Keywords- **CPU scheduling, operating system, scheduling algorithm**.

I. INTRODUCTION

Scheduling is one of the fundamental functions of any operating system, since almost all computer resources are scheduled before use. The CPU is of course, one of the primary computer resources [1]. Thus its scheduling algorithm is heart of the OS design. When more than one process is run-able, the OS must decide which one is to run first. That part of the OS concerned with this decision is called scheduler and the algorithm it uses is called scheduling algorithm. A CPU scheduler is the part of an operating system and responsible for arbitrating access to the CPU.

OS may feature up to 3 distinct types of schedulers: a long-term scheduler (also known as an admission scheduler or high-level scheduler), a mid-term or medium-term scheduler and a short-term scheduler (also known as a dispatcher or CPU scheduler).

A. Long-term Scheduler

The long-term or admission scheduler decides which jobs or processes are to be admitted to the ready queue; that is, when an attempt is made to execute a process its admission to the set of currently executing processes is either authorized or delayed by the long-term scheduler. Thus, this scheduler dictates what processes are to run on a system, and the degree of concurrency to be supported at any one time.

B. Mid-term Scheduler

The mid-term scheduler temporarily removes process from main memory and place them on secondary memory (such as a disk drive) or vice versa. This is commonly referred to as "swapping of processes out" or "swapping in" (also incorrectly as "paging out" or "paging in").

C. Short-term Scheduler

The short-term scheduler (also known as the CPU scheduler) decides which of processes in the ready queue, in-memory are to be executed (allocated a CPU) next following a clock interrupt, an Input-Output (IO) interrupt and an OS call or another form of signal. Thus the short-term scheduler makes scheduling decisions much more frequent than the long-term or mid-term schedulers. This scheduler can be pre-emptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non pre-emptive (also known as "voluntary" or "co-operative"), in that case the scheduler is unable to force processes off the CPU.

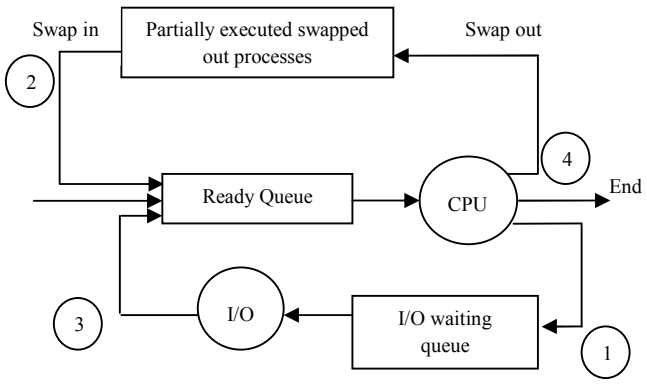

Fig. 1. Process of Schedulers

Fig. 1. Shows the following states have been executed in the CPU Scheduler.

1. When a process switches from the running state to the waiting state.
2. When a process switches from the running state to the ready state.
3. When a process switches from the waiting state to the ready state.
4. When a process terminates.

The success of a CPU scheduler depends on the design of high quality scheduling algorithm. High-quality CPU scheduling algorithms rely mainly on criteria such as CPU

978-0-7695-4058-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ACE.2010.14

utilization rate, throughput, turnaround time, waiting time and response time. Thus, the main impetus of this work is to develop a generalized optimum high quality scheduling algorithm suited for all types of job.

The rest of the paper is organised as follows. Section II, gives review on scheduling algorithms. The proposed algorithm, Results and Discussion have been given in section III and IV, followed by conclusion and future work.

II. Scheduling Algorithms

A. Algorithms and Its Characteristics

The fundamental scheduling algorithms and its characteristics are described in this section.

a. First Come First Serve

The most intuitive and simplest technique is to allow the first process submitted to run first. This approach is called as first-come, first-served (FCFS) scheduling. In effect, processes are inserted into the tail of a queue when they are submitted [2]. The next process is taken from the head of the queue when each finishes running.

i. Algorithm

Step 1: The first requested process is allocated to the CPU first.
Step 2: The new processes are added at the tail of the ready queue.
Step 3: When the process terminates, dequeue the next process from head of ready queue and run it.

ii. Characteristics

- The lack of prioritization does permit every process to eventually complete, hence no starvation.
- Turnaround time, waiting time and response time is high.
- One Process with longest burst time can monopolize CPU, even if other process burst time is too short. Hence, throughput is low [3].

b. Shortest Job First

The process is allocated to the CPU which has least burst 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 [2]. This algorithm is designed for maximum throughput in most scenarios.

i. Algorithm

Step 1: Allocate the CPU to the process that has the shortest burst time.
Step 2: If one or more process has the same burst time
{
Allocate the CPU to the process according to the FCFS scheduling
}

ii. Characteristics

- The real difficulty with the SJF algorithm is, to know the length of the next CPU request.
- SJF minimizes the average waiting time [3] because it services small processes before it services large ones. While it minimizes average wait time, it may penalize processes with high service time requests. If the ready list is saturated, then processes with large service times tend to be left in the ready list while small processes receive service. In extreme case, when the system has little idle time, processes with large service time will never be served. This total starvation of large processes is a serious liability of this algorithm.

c. Round Robin

The Round Robin (RR) scheduling algorithm assigns a small unit of time, called time slice or quantum. The ready processes are kept in a queue. The scheduler goes around this queue, allocating the CPU to each process for a time interval of assigned quantum. New processes are added to the tail of the queue [4].

i. Algorithm

Step 1: Choose the time quantum and assign it for each process.
Step 2: Allocate the CPU to the process according to the FCFS scheduling.
Step 3: If (burst time of the process < time quantum).
{
Allocate the CPU to that process till it terminates.
}
Else
{
The process will occupy the CPU till the time quantum and it is added to the tail of the ready queue for the next round of execution.
}

ii. Characteristics

- Setting the quantum too short causes too many context switches and lower the CPU efficiency.
- Setting the quantum too long may cause poor response time and approximates FCFS.
- Because of high waiting times, deadlines are rarely met in a pure RR system.

d. Priority Scheduling

The O/S assigns a fixed priority rank to each process. Lower priority processes get interrupted by incoming higher priority processes.

i. Algorithm

Step 1: Assign priority to each of the processes in the ready queue.

Step 2: Allocate the CPU to the process that has the highest priority and so on.

Step 3: If two or more process has equal-priority then

{

Allocate the CPU to the process according to the FCFS.

}

ii. Characteristics

- Starvation can happen to the low priority process.
- The waiting time gradually increases for the equal priority processes [5].
- Higher priority processes have smaller waiting time and response time.

B. Computation of Gantt chart, Waiting Time and Turnaround Time

Consider the following set of processes, with the length of the CPU-burst time in milliseconds is shown in Table. 1:

TABLE 1. PROCESSES WITH ITS ID AND BURST TIME

Process ID	Burst Time(ms)
P_1	10
P_2	1
P_3	2
P_4	1
P_5	5

a. First Come First Serve

i. Gantt Chart

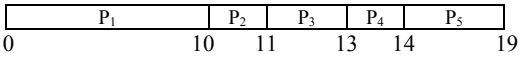

Fig. 2 Gantt chart for FCFS

b. Shortest Job First

ii. Gantt Chart

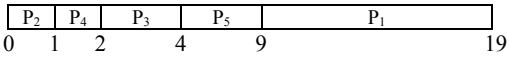

Fig. 3 Gantt chart for SJF

c. Round Robin

Assign time quantum as 5 ms for each process.

i. Gantt Chart

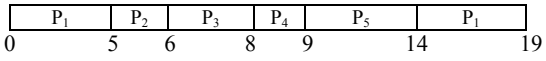

Fig. 4 Gantt chart for RR

d. *Priority Scheduling*

Priority is assigned for each process as follows.

TABLE. 2. PROCESSES WITH ITS ID, BURST TIME AND PRIORITY

Process ID	Priority
P_1	3
P_2	1
P_3	3
P_4	4
P_5	2

i. Gantt Chart

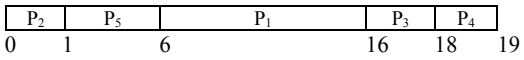

Fig. 5 Gantt chart for PS

For example, turnaround time for the process is calculated as time of submission of a process to the time of completion of the process is obtained through Gantt chart for SJF scheduling. Turnaround time for process P_1, P_2, P_3, P_4 & P_5 is observed as 19, 1, 4, 2 & 9 respectively and average turnaround time is (19+1+4+2+9)/5=7ms.

The waiting time for the process is calculated as time taken by the process to wait in the ready queue is observed from Gantt chart for SJF scheduling. Waiting time for process P_1, P_2, P_3, P_4 & P_5 is obtained as 9, 0, 1, 2 & 4 respectively and average waiting time is (9+0+1+2+4)/5=3.2ms.

Similarly the turnaround time and waiting time is calculated for all other algorithms and summarized in Table 3 and Table 4.

TABLE. 3. WAITING TIME FOR INDIVIDUAL PROCESS AND AVERAGE WAITING TIME FOR EACH SCHEDULING.

Process ID	WAITING TIME (ms)			
	FCFS	SJF	Round Robin	Priority Scheduling
P_1	0	9	9	6
P_2	10	0	5	0
P_3	11	2	6	16
P_4	13	1	8	18
P_5	14	4	9	1
Avg Waiting Time	9.6	3.2	7.4	8.2

TABLE. 4. TURN AROUND TIME FOR INDIVIDUAL PROCESS AND AVERAGE TURNAROUND TIME FOR EACH SCHEDULING

Process ID	TURNAROUND TIME (ms)			
	FCFS	SJF	Round Robin	Priority Scheduling
P_1	10	19	19	16
P_2	11	1	6	1
P_3	13	4	8	18
P_4	14	2	9	19
P_5	19	9	14	6
Avg Turnaround Time	13.4	7	11.2	12

From the above discussion it is clear that First Come First Serve (FCFS) & Shortest Job First (SJF) is generally suitable for batch operating systems and Round Robin (RR) & Priority Scheduling (PS) is suitable for time sharing systems. No algorithm is optimum for all type of jobs. Hence it is necessary to develop an algorithm with an optimum criteria and suitable for all scenarios.

III. Proposed Scheduling Algorithm

A. Proposed Algorithm and its Characteristics

The proposed algorithm it combines the working principle of fundamental scheduling algorithms. The algorithm is as follows:

Step 1: Calculate the time quantum as follows.

$$\text{Time quantum} = \sum_{i=1}^{n} P_i \,/\, n \qquad (1)$$

where P_i is the burst time of Process $_i$,
n is the number of process.

Step 2: Assign the time quantum and apply for each process.

Step 3: Shuffle the processes in ascending order in the ready queue such that the head of the ready queue contains the lowest burst time.

Step 3: If one or more process has equal burst time then
{
Allocate the CPU to the processes according to First Come basis.
}

Step 4: If (burst time of the process < time quantum).
{
Allocate the CPU to that process till it terminates.
}
Else
{
The process will occupy the CPU till the time quantum and it is added to the tail of the ready queue for the next round of execution.
}

Characteristics

- The starvation of long process can be avoided by giving a time quantum for each.
- No process can monopolize CPU.
- CPU utilisation, waiting time, turnaround time, response time and throughput can be optimum.

B. Computation of Gantt Chart, Waiting Time and Turnaround Time

i. Calculation of time quantum

By using (1) the time quantum is calculated as follows.

Time quantum= (10+1+2+1+5)/5=3.8ms ≅ 4 ms

ii. Gantt Chart

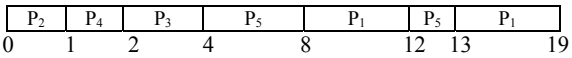

Fig. 6 Gantt chart for Proposed Algorithm

The average turnaround time and average waiting time is calculated as follows.

Average waiting time = (9+0+2+1+8)/5 = 4 ms.

Average turnaround time=(19+1+4+2+13)/5=7.8ms.

TABLE 5. Waiting Time and Turnaround Time for Proposed Algorithm

Process ID	Proposed Algorithm	
	Waiting Time (ms)	Turnaround Time (ms)
P_1	9	19
P_2	0	1
P_3	2	4
P_4	1	2
P_5	8	13
Average	4	7.8

IV. Results and Discussion

The proposed algorithm has been simulated and justified with C++ code; it can be an innovative move in case of CPU scheduling and can be easily implemented in the OS. Comparison of the proposed algorithm with the existing algorithm is shown in Fig. 7 and Fig. 8.

It is clearly observed the turnaround time, waiting time and response time of the processes are optimum for proposed algorithm compared to all other fundamental algorithms from Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6, (Fig 1 to Fig 6 are Gantt Chart) Fig. 7 and Fig. 8. It can also be observed that throughput and CPU utilisation rate are optimum.

i. Waiting Time

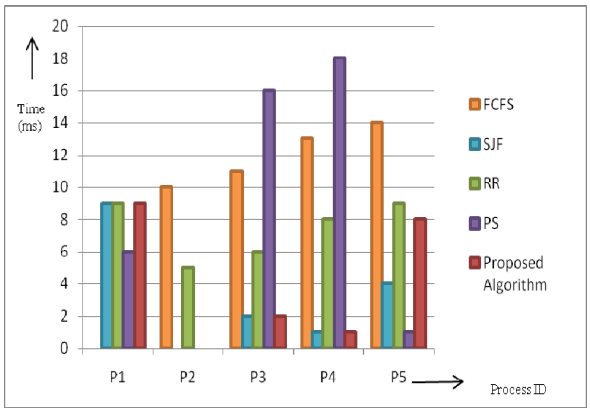

Fig. 7 Comparison of proposed algorithm with fundamental algorithm-Waiting Time

ii. Turnaround Time

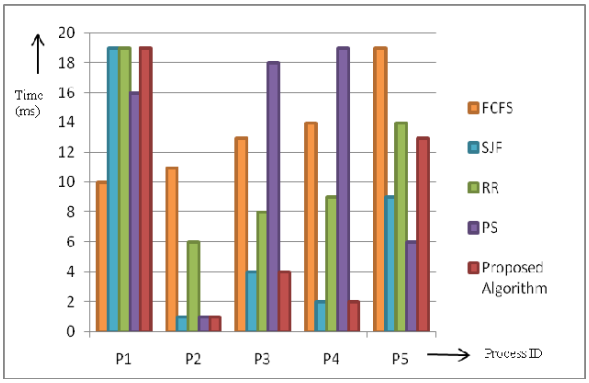

Fig. 8 Comparison of proposed algorithm with fundamental algorithm-Turnaround Time

V. CONCLUSION AND FUTURE WORK

In this work, the proposed algorithm to serve all types of job with optimum scheduling criteria. The treatment of shortest process in SJF scheduling tends to result in increased waiting time for long processes. And the long process will never get served, though it produces minimum average waiting time and average turnaround time. This problem can be overcome by the proposed algorithm.

It is recommended that any kind of simulation for any CPU scheduling algorithm has limited accuracy. The only way to evaluate a scheduling algorithm to code it and has to put it in the operating system, only then a proper working capability of the algorithm can be measured in real time systems. Hence in future the proposed algorithm will be implemented and can be tested in open source (LINUX).

ACKNOWLEDGEMENT

We feel highly elated to render our heartful thanks to our beloved Chairman Thiru V. G. Rajendran and Thirumathy Indira Rajendran for having provided us a grandeur infrastructure which creates a congenial atmosphere to work on this topic. It is our sole duty to render our gratitude to our respected Principal Dr. N. Vasudevan who stimulated and motivated us constantly to take up this great task of working on this topic. We gladly thank Ms. R. Padmini for the assistance in enriching the content.

REFERENCES

1. Sukanya Suranauwarat, "A CPU Scheduling Algorithm Simulator", October 10-13, 2007, Milwaukee, WI 37th ASEE/IEEE Frontiers in Education Conference.
2. Abraham Silberschatz, Peter Baer Galvin, Greg Gangne;"Operating System Concepts", edition-6, 2002, John Wiley and Sons, INC.
3. Mr. Umar Saleem and Dr. Muhammad Younus Javed, "Simulation of CPU Scheduling Algorithm", 0-7803-6355-8/00/ $10.00 @ 2000 IEEE.
4. Sun Huajin', Gao Deyuan, Zhang Shengbing, Wang Danghui;"Design Fast Round Robin Scheduler in FPGA", 0-7803-7547-5/021/$17.00 @ 2002 IEEE.
5. Md. Mamunur Rashid and Md. Nasim Adhtar; "A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 2036-2039,2009.