© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

Comparison Study of Processor Scheduling Algorithms

Vibhu Chinmay, Shubham Sachdeva
Student, B.Tech, Department of Electronics and Computers Engineering
Dronacharya College of Engineering, Gurgaon, India

Abstract: **Operating system efficiency is affected by several factors. Scheduling of CPU is one of the critical factors that affect the efficiency. In case of process scheduling we allocate processes to processor in a specified sequence so that efficiency of the system is maximized. Developing CPU scheduling algorithms and understanding their impact in practice can be difficult and time consuming due to the need to modify and test operating system kernel code and measure the resulting performance on a consistent workload of real applications. As processor is the important resource, CPU scheduling becomes very important in accomplishing the operating system (OS) design goals. The intention should be allowed as many as possible running processes at all time in order to make best use of CPU. This paper presents a state diagram that depicts the comparative study of various scheduling algorithms for a single CPU and shows which algorithm is best for the particular situation. Using this representation, it becomes much easier to understand what is going on inside the system and why a different set of processes is a candidate for the allocation of the CPU at different time. The objective of the study is to analyze the high efficient CPU scheduler on design of the high quality scheduling algorithms which suits the scheduling goals..**

Keywords: Scheduler, State Diagrams, CPU-Scheduling, Performance

I. INTRODUCTION

In a single-processor syste*m, 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 [1]. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system. There are a number of scheduling algorithms based on which we can evaluate the various scheduling criteria viz. avg. waiting time, turnaround time etc. Among them First-Come First-Served (FCFS) Scheduling, Shortest-Job-First (SJF) Scheduling, Priority Scheduling, Round Robin (RR) Scheduling, Multilevel Queue Scheduling are of much importance and are widely used for scheduling of jobs in a processor.

II. CPU SCHEDULER

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 preemptive, 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.

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 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.

III. SCHEDULING CRITERIA

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. 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. The criteria include the following:

- **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 each process gets a fair share of the CPU.

IV. SCHEDULING ALOGRITHMS

A. Algorithm 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. For this algorithm the ready queue is maintained as a FIFO queue. PCB (Process Control Block) of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue. This idea is illustrated in the four state diagram of figure 1.

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.

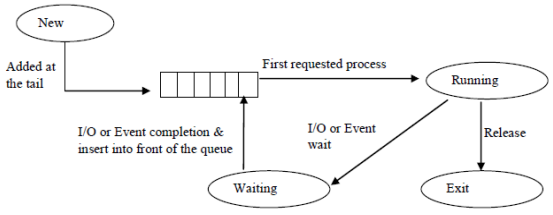

Figure 1: First Come First Serve Scheduling

b. *Non preempted Shortest Job First*

For this algorithm the ready queue is maintained in order of CPU burst length, with the shortest burst length at the head of the queue. The process is allocated to the CPU which has least burst time. A

PCB of a process submitted to the system is linked to the queue in accordance with its CPU burst length. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. When a process has completed its task it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.This algorithm is designed for maximum throughput in most scenarios. This idea is illustrated in the four state diagram of figure 2.

Characteristics

- The real difficulty with the SJF algorithm is, to know the length of the next CPU request.
- SJF minimizes the average waiting time 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.

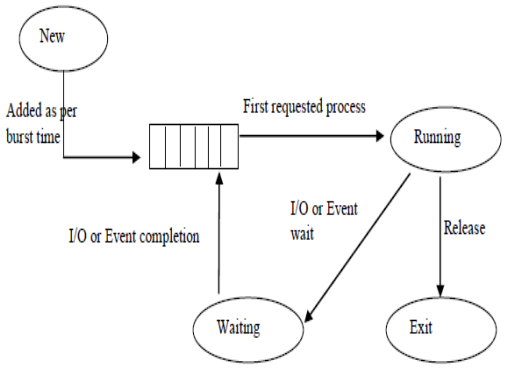

Figure 2: Shortest Job First Scheduling

c. *Round Robin*

The Round Robin (RR) scheduling algorithm assigns a small unit of time, called time slice or quantum. For this algorithm the ready queue is maintained as a FIFO queue. A PCB of a process submitted to the system is linked to the tail of the queue. The algorithm dispatches processes from the head of the ready queue for execution by the CPU. Processes being executed is preempted on expiry of a time quantum, which is a system defined variable. A pre-empted process's PCB is linked to the tail of the ready queue. When a process has completed its task, i.e. before the expiry of the time quantum, it terminates and is deleted from the system. The next process is then dispatched from the head of the ready queue.This idea is illustrated in the four state diagram of figure 3.

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.

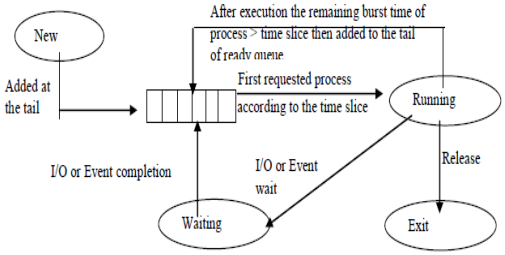

Figure 3: Round Robin Scheduling

d. *Priority Scheduling*

The O/S assigns a fixed priority rank to each process. Lower priority processes get interrupted by incoming higher priority processes. In this algorithm, priority is associated with each process and on the basis of that priority CPU is allocated to the processes. Higher priority processes are executed first and lower priority processes are executed at the end. If multiple processes having the same priorities are ready to execute, control of CPU is assigned to these processes on the basis of FCFS. Priority Scheduling can be preemptive and non-preemptive in nature.This idea is illustrated in the four state diagram of figure 4.

Characteristics

- Starvation can happen to the low priority process.

- The waiting time gradually increases for the equal priority processes.
- Higher priority processes have smaller waiting time and response time.

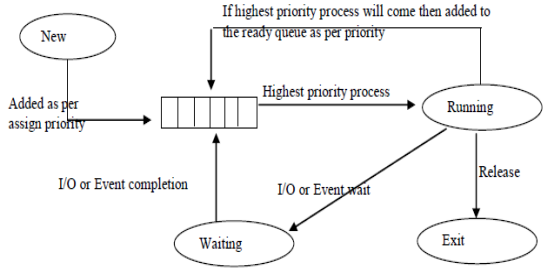

Figure 4: Priority Scheduling

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:

Process ID	Burst Time(ms)
P0	12
P1	2
P2	3
P3	2
P4	6

Table 1: Processes with Its Id and Burst Time

a. First Come First Serve

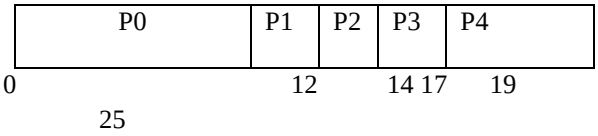

Figure 5: Gantt chart for FCFS

b. Shortest Job First

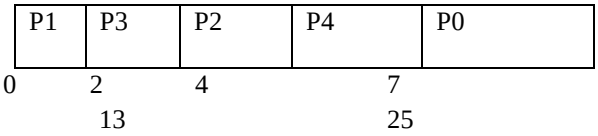

Figure 6: Gantt chart for SJF

c. Round Robin

Assign time quantum as 5 ms for each process.

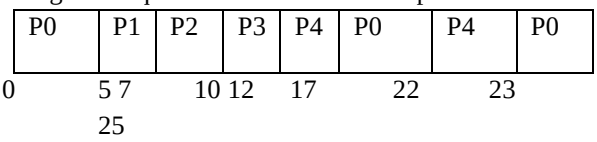

Figure 7: Gantt chart for Round Robin

d. Priority Scheduling

Priority is assigned for each process as follows:

Process ID	Burst Time(ms)	Priority
P0	12	3
P1	2	1
P2	3	3
P3	2	4
P4	6	2

Table 2: Processes with Its Id, Burst Time and Priority

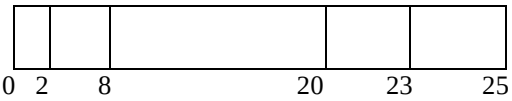

Figure 8: Gantt chart for Priority Scheduling

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 P0, P1, P2, P3 & P4 is observed as 25,2,7,4 & 13 respectively and average turnaround time is (25+2+7+4+13)/5=10.2 ms.

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 P0, P1, P2, P3 & P4 is obtained as 13, 0, 4, 2 & 7 respectively and average waiting time is (13+0+4+2+7)/5=5.2ms.

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.

V. CONCLUSION & FUTURE WORK

This paper mainly emphasizes on the comparison criteria of different CPU Scheduling algorithms.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. 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.

REFERENCES

1. Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India,
2. Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F. Safaei, (2005). Processing resource scheduling in programmable networks, Computer communication, 28:676-687
3. Sun Huajin', Gao Deyuan, Zhang Shengbing, Wang Danghui; " Design fast Round Robin Scheduler in FPGA", 0-7803-7547-5/021/$17.00@2002 IEEE
4. Md. Mamunur Rashid and Md. Nasim Adhtar, " A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 2036-2039,2009
5. Sukanya Suranauwarat, "A CPU Scheduling Algorithm Simulator", October 10-13, 2007, Milwaukee, WI 37th ASEE/IEEE Frontiers in Education Conference.
6. Andrew S. Tanenbaum, Albert S. Woodhull, "Operating Systems Design and Implementation", Second Edition
7. Milan Milenkovic, "Operating System Concepts and Design", McGRAW-HILL, Computer Science Series, Second Edition.