Lecture 4- CPU Scheduling

Instructor : Bibhas Ghoshal (bibhas.ghoshal@iiita.ac.in)

Autumn Semester, 2019

Lecture Outline

- CPU Scheduling Basics
- CPU Scheduler and Dispatcher
- Scheduling Criteria
- First Come First Serve (FCFS) Scheduling
- First Shortest Job First (SJF) Scheduling

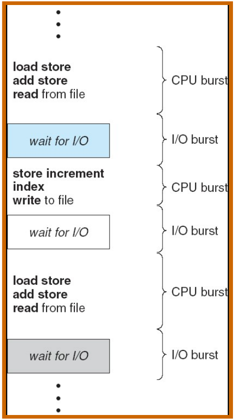
References and Illustrations have been used from:

- lecture slides of the book - Operating System Concepts by Silberschatz, Galvin and Gagne, 2005
- Modern Operating System by Andrew S. Tanenbaum
- lecture slides of CSE 30341: Operating Systems (Instructor : Surendar Chandra),

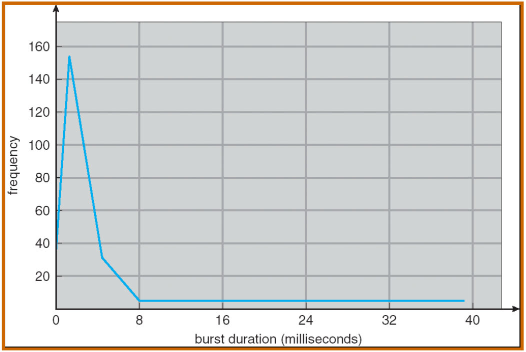
CPU Scheduling: Basic Concepts

- Maximum CPU utilization obtained with multiprogramming - several processes are kept in memory, while one is waiting for I/O, the OS gives the CPU to another process
- CPU scheduling depends on the observation that processes cycle between CPU execution and I/O wait.

Alternating Sequence of CPU And I/O Bursts



Histogram of CPU Burst Times



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state (e.g. I/O request)
  2. Switches from running to ready state (e.g. Interrupt)
  3. Switches from waiting to ready (e.g. I/O completion)
  4. Terminates
- Scheduling under 1 and 4 is *non-preemptive* (cooperative)
- All other scheduling is *preemptive* - have to deal with possibility that operations (system calls) may be incomplete

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  1. switching context
  2. switching to user mode
  3. jumping to the proper location in the user program to restart that program
- Dispatch latency – time it takes for the dispatcher to stop one process and start another running
  - Should be as low as possible

Scheduling Criteria

- CPU utilization (max) – keep the CPU as busy as possible
- Throughput (max) – # of processes that complete their execution per time unit
- Turnaround time (min) – amount of time to execute a particular process
- Waiting time (min) – amount of time a process has been waiting in the ready queue
- Response time (min) – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)
- In typical OS, we optimize each to various degrees depending on what we are optimizing the OS

Optimization Criteria

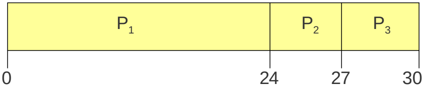
- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time
- Analysis using Gantt chart (illustrates when processes complete)

First Come First Serve (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order: P_1 , P_2 , P_3

The Gantt Chart for the schedule is:



Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

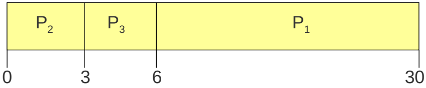
Average waiting time: $(0 + 24 + 27)/3 = 17$

First Come First Serve Scheduling

Suppose that the processes arrive in the order

P_2 , P_3 , P_1

The Gantt chart for the schedule is:



Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$

Average waiting time: $(6 + 0 + 3)/3 = 3$

Much better than previous case

Convoy effect short process behind long process

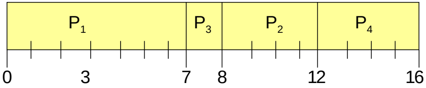
Shortest Job First (SJF)

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest times
- Two schemes:
  - *nonpreemptive* – once CPU given to the process, it cannot be preempted until completes its CPU burst
  - *preemptive* – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the **Shortest-Remaining-Time-First (SRTF)**
- SJF is optimal – gives minimum average waiting time for a given set of processes

Non-preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

SJF (non-preemptive)

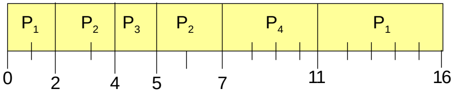


Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

SJF (preemptive)

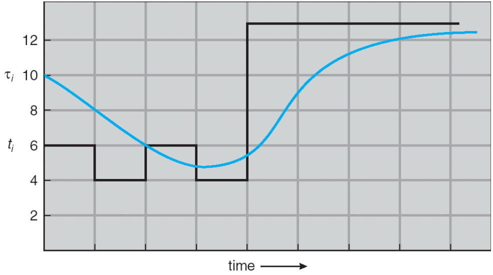


Average waiting time = (9 + 1 + 0 +2)/4 = 3

Determining Length of Next CPU Burst

- Can only estimate the length
- Can be done by using the length of previous CPU bursts, using exponential averaging
  1. t_n = actual length of n^{th} CPU burst
  2. τ_{n+1} = predicted value of next CPU burst
  3. α = $0 \leq \alpha \leq 1$
  4. $\tau_{n+1} = \alpha\tau_n + (1 - \alpha)\tau_n$

Prediction of the Length of the Next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Examples of Exponential Averaging

- if $\alpha = 0 \implies \tau_{n+1} = \tau_n$: Recent history does not count
- if $\alpha = 1 \implies \tau_{n+1} = \alpha \times t_n$: Only the actual last CPU burst counts