CS341: Operating System

Process Scheduling

Lect11 : 26th Aug 2014

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

1

Outline

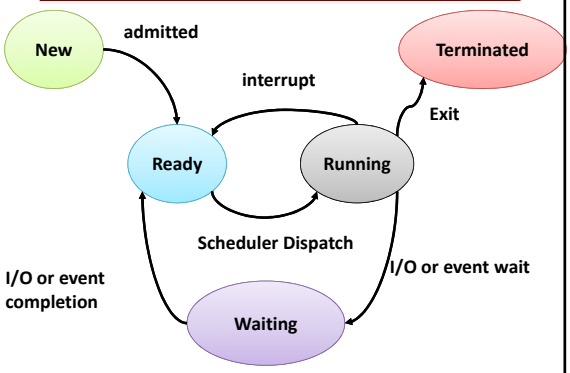
- Process Concepts Recap
  - Process States, PCB
  - Context Switch
  - **Job Queue, Ready Queue, De**
- Scheduling
  - **System Oriented**
  - **Theoretical Analysis**

2

Recap

3

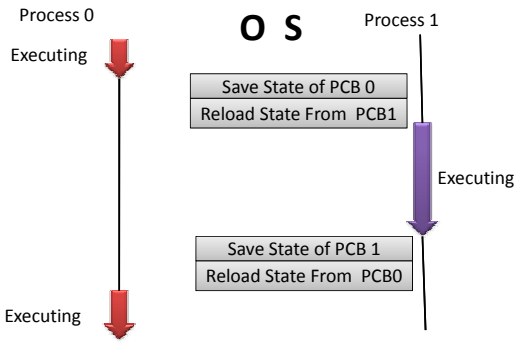
Process State: State Diagram



PCB: Info. related to Process

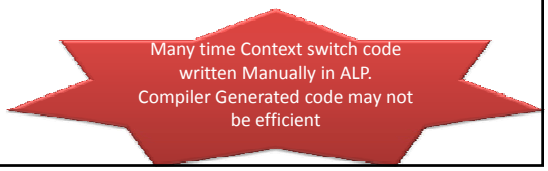
- Process Info.
  - Process state, PC, Registers
- Other info
  - CPU scheduling info. priorities, scheduling queue pointers
  - Mem-mngt info: memory allocated to the process
  - Acc. info. – CPU used, time since start, time limits
  - I/O status info. – I/O dev allocated to process, list of open files

CPU Switch From Process to Process



Context Switch

- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB ➔ the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU ➔ multiple contexts loaded at once



Process Scheduling

- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes **migrate among the various queues**

Next…

9

Process Scheduling

10

Scheduling Evaluation Metrics

- CPU utilization
  - Percentage of time the CPU is not idle
- Throughput
  - Completed processes per time unit
- Turnaround time
  - Submission to completion

Scheduling Evaluation Metrics

- Waiting time
  - time spent on the ready queue
- Response time
  - response latency
  - "response time" most important for interactive jobs (I/O bound)
- Predictability
  - variance in any of these measures

The right evaluation metric depends on the context

Scheduling Algorithm Optimization Criteria

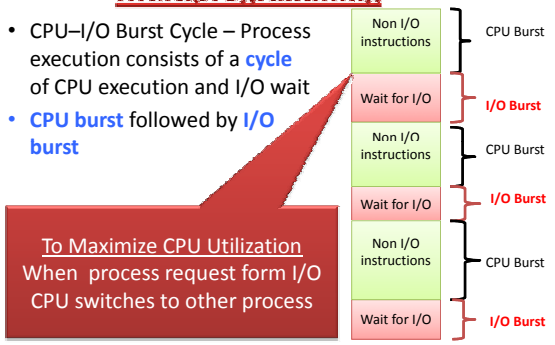
- Maximize
  - CPU utilization
  - Throughput
- Minimize
  - Turnaround time
  - Waiting time
  - Response time

"The perfect CPU scheduler"

- Minimize latency
  - Response or job completion time
- Maximize throughput
  - Maximize jobs / time.
- Maximize utilization: keep I/O devices busy.
  - Recurring theme with OS scheduling
- Fairness: everyone makes progress, no one starves

Max CPU Util. obtained with Multiprogramming

- CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**



To Maximize CPU Utilization
When process request form I/O CPU switches to other process

Assumption

- Task/Process/Job used interchangbly
- Let all tasks in memory
- Let all tasks don't do any IPC
- All task are independent
- All tasks don't require any other resource other then CPU

Five Popular Scheduling

- First Come First Serve
- Shortest Job First
- Shortest Remaining Time First
  - SJF-I
- Round-Robin Scheduler
- Priority Scheduler
- Priority-I
- Multi-Level Priority Queue
- Feed Back Priority Queue

18

Flow Time of a Job : Turn Around Time

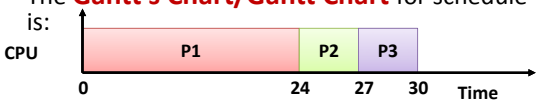
- $F_i = C_i - A_i$
  - C_i= completion time, A_i=Arrival time
  - Flow time depended on both C_i and A_i, If A_i=0, depends on C_i
  - In uniprocessor: $F_i = W_i$ the waiting time + CC_i Compute time
- Average Flow time = $F_i' = \sum F_i$
  - In uniprocessor : Minimize Agv Flow Time = Mimimize Avg Waiting time
  - Min(F_i') =Min($\sum W_i$) as CPU is completely busy for all the time
- Minimize Weighted Flow time: Priority Version

19

FCFS Scheduling

Process	Burst Time/ Execution Time
P1	24
P2	3
P3	3

- Assume: processes arrive in the order: P_1 , P_2 , P_3 The **Gantt's Chart/Gantt Chart** for schedule is:



- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

FCFS

Convoy effect

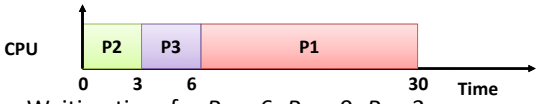
short process behind long process

21

FCFS Scheduling

Process	Burst Time/ Execution Time
P1	24
P2	3
P3	3

- Assume: 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

Problem: Cook of Restaurant

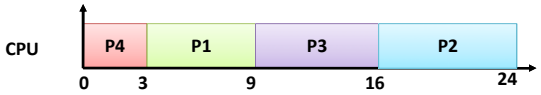
- You work as a short-order cook
  - Customers come in and specify which dish they want
  - Each dish takes a different amount of time to prepare
- Your goal:
  - Minimize average time the customers wait for their food
- What strategy would you use ?
  - Note: most restaurants use FCFS.

Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- **SJF is optimal** – gives minimum average waiting time for a given set of processes
  - **The difficulty is knowing the length of the next CPU request**
  - Could ask the user

Example of SJF

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

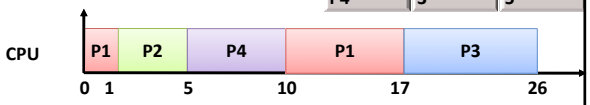


- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

SJF-Preemptive: Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis
- **Dynamic Decision @Runtime**

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

Two Processor System

- **Minimize Agv Flow Time ≠ Mimimize Avg Waiting time**
- C_{max} is not equal for both the processor
- Example
  - Six Process: P1=1, p2=1, p3=1, p4=2, p5=2, p6 =3
  - Two Processors : C1 and C2
  - Minimize Cmax= partition {1,1,1,2,2,3} in to two set S1 and S2 such that their sum is equal or almost equal
  - SOL1: {1,1,1,2}{2,3} SOL2: {1,2,2},{1,1,3}

27

Two Processor System

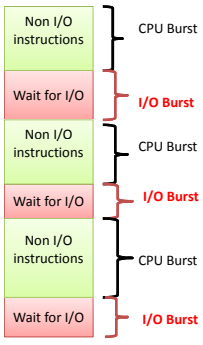
- Two set Partition problem is intractable problem
  - Suppose dividing {t1,t2, t3, …tn} into two set S1 and S2 such that Sum{S1}≈Sum{S2}
- Decision version
  - Pseudo Polynomial solution, Dynamic Programming
- Approach
  - Approximation
  - Karmarkar Approach

http://en.wikipedia.org/wiki/Partition_problem

28

Determining Length of Next CPU Burst

- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst
- Can be done by using the length of previous CPU bursts, using exponential averaging



Prediction: Next CPU Burst

- Can be done by using the length of previous CPU bursts, using exponential averaging
  - t_n is actual CPU Burst of nth CPU Burst
  - τ_{n+1} = predicted values of next cpu burst

$$t_3 = f(t_1,t_2,t_3,\tau_1,\tau_2,\tau_3);$$

$$= \text{may be } f(t_3,\tau_3);$$

Another funny Example: http file transfer

BW depends on chunk size C_n, Initial = 1+x, $C_n=(1+x)^n$ where x<1

Prediction: Next CPU Burst

- α, 0 ≤ α ≤ 1

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$$

- Commonly, α set to ½
- Preemptive version called **shortest-remaining-time-first**

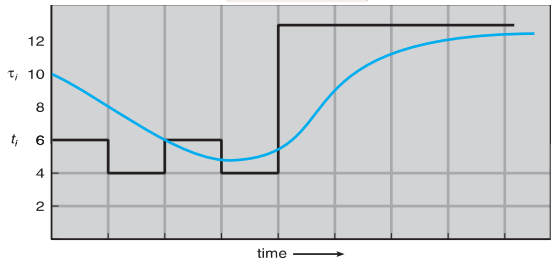
Examples of Exponential Averaging

- $\alpha = 0$: $\tau_{n+1} = \tau_n$, Recent history does not count
- $\alpha = 1$: $\tau_{n+1} = \alpha\, t_n$, Only actual last CPU burst counts
- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1-\alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1-\alpha)^{n+1} \tau_0$$

- Since both α and (1 - α) are less than or equal to 1, each successive term has less weight than its predecessor

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	...