CAP. 5 : CPU SCHEDULING

Concetti di base: multiprogrammazione

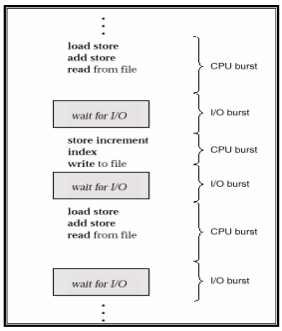
Da ora in poi sul libro di testo…

Multiprogrammazione = Multitasking

… e cioe' ***piu' di un processo*** alla volta pronto ad essere eseguito viene tenuto in memoria centrale e ***condivide il tempo di CPU*** con tutti gli altri

Concetti di base: CPU burst e I/O burst

Il comportamento tipico di un processo e' di ***alternare*** una sequenza di ***continuo uso di CPU*** (CPU burst) con una sequenza di ***operazioni di I/O*** (I/O burst)

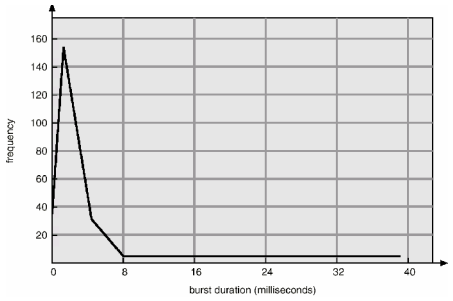


Quindi…

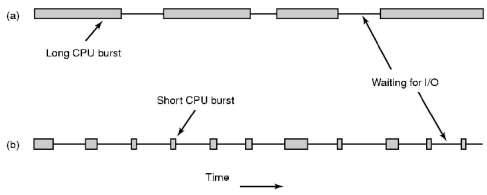
… la possibilita', durante l'***I/O burst*** di un processo, di mandare in esecuzione qualche altro processo (***multiprogrammazione***) porta ad una piu' alta utilizzazione della ***preziosa risorsa CPU***

L'andamento dei cicli CPU-I/O burst influenza le ***scelte di gestione dei processi***, effettuate dallo ***scheduler***

Quanto dura un CPU burst?!



Reale andamento ma fittizi valori di tempo!



a) Un processo CPU-bound
b) Un processo I/O bound

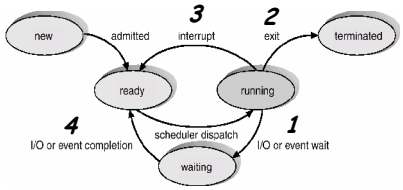
CPU Scheduler: cosa fa

Quando necessario seleziona, tra i processi in memoria che sono pronti (cioe' nella ***ready queue***) quello che deve ***andare in esecuzione***

Quando necessario?

1. Un processo va da running a waiting
2. Un processo in esecuzione termina
3. Un processo va da running a ready

4. Un processo va da waiting a ready (?!)



Dispatcher

• Lo ***scheduler*** sceglie il processo da eseguire

• Il ***dispatcher*** da' il controllo della CPU al processo scelto, e cioe':

- context switching
- salto alla locazione propria del programma utente scelto per farlo ripartire
- switching a user mode

• ***Dispatch latency*** – tempo impiegato dal dispatcher per svolgere il suo compito

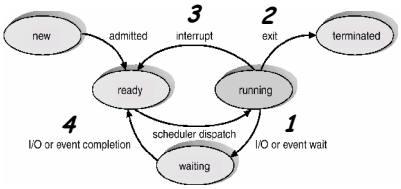
Concetti di base: preemption…

Qualsiasi ***azione*** di scheduling si dice ***preemptive*** se ha ***la possibilita' di interrompere il processo in esecuzione*** per sostituirlo con un altro

Un'azione di scheduling e' quindi ***non-preemptive*** quando ***attende che il processo*** in esecuzione ***abbandoni autonomamente la CPU*** comunque prima di agire

… e qualche considerazione

le azioni conseguenti a 1 e 2 sono *non-preemptive*
le azioni conseguenti a 3 e 4 sono *preemptive*



Inconsistenza di dati condivisi – a livello utente oppure, ancor piu' grave, su strutture dati di kernel

Da azione a politica di scheduling

Una ***politica*** di scheduling e' un algoritmo che ***regola ogni azione*** di scheduling e quindi ***decide quando*** e ***come*** selezionare un processo da eseguire

Separazione meccanismo-politica:

Meccanismo → scheduler
Politica → scheduling

Parametri di confronto tra politiche : a livello di OS

- ***Utilizzazione della CPU*** – percentuale di tempo in cui la CPU e' occupata a fare qualcosa
- ***Throughput*** – numero di processi che completano la loro esecuzione per unita' di tempo

Da massimizzare

Parametri di confronto tra politiche : a livello di processo

- ***Turnaround time*** – tempo necessario ad eseguire un processo
- ***Tempo di attesa*** – tempo totale di attesa di un processo nella ready queue ***(parametro che piu' direttamente e' influenzato dalla politica!)***
- ***Tempo di risposta*** – tempo che intercorre tra una richiesta e l'arrivo della prima risposta (non il completamento dell'output del processo)

Da minimizzare

… e qualche variazione

Potrebbe essere preferibile ottimizzare i ***valori estremi*** di un parametro piuttosto che i medi, per esempio:

invece di tendere a minimizzare il ***tempo di attesa*** medio si potrebbe puntare a ***minimizzare quello massimo***…

… oppure si potrebbe ***minimizzare la varianza*** di un dato parametro

Politiche che consideriamo

1. First Come First Served
2. Shortest Job First
3. Con priorita'
4. Round-Robin
5. Code multilivello (con feedback)

Parametro di confronto :
tempo di attesa medio

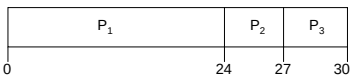
Orizzonte di osservazione :
1 CPU burst per processo

1. First-Come, First-Served (FCFS)…

I processi vengono schedulati nello stesso ordine con il quale arrivano nella ready queue

… un esempio…

Processo	Burst Time
P_1	24
P_2	3
P_3	3



- Tempo di attesa per processo:

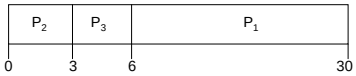
  $P_1 = 0; P_2 = 24; P_3 = 27$

- Tempo di attesa medio: $(0 + 24 + 27)/3 = 17$

… ed una variazione

Supponiamo invece che i processi siano arrivati nel seguente ordine:

P_2 , P_3 , P_1



- Tempo di attesa per processo:

  $P_1 = 6; P_2 = 0; P_3 = 3$

- Tempo di attesa medio: $(6 + 0 + 3)/3 = 3$

Convoy effect: processi brevi dietro a processi lunghi

2. Shortest-Job-First (SJF)…

Schedula il processo che presenta il prossimo CPU burst piu' breve

… in realta' si tratta di ***shortest next CPU burst!***

Solo nei batch si puo' stimare il tempo di un intero job

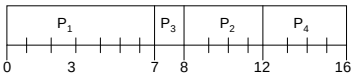
… due possibilita'

Non-preemptive – una volta che la CPU e' assegnata ad un processo non gliela si puo' togliere finche' non completa il suo CPU burst

Preemptive – se un nuovo processo arriva nella ready queue con ***un burst piu' corto del tempo rimanente*** del corrente processo allora lo si rimpiazza (Shortest-Remaining-Time-First, ***SRTF***)

Non-preemptive (esempio)

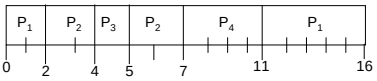
Processo	Istante di arrivo	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Tempo medio di attesa: (0 + 6 + 3 + 7)/4 = 4

Preemptive (esempio)

Processo	Istante di arrivo	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Tempo medio di attesa: (9 + 1 + 0 +2)/4 = 3

SJF e' ottimale, cioe' da' il minimo tempo medio di attesa per ogni dato insieme di processi!!!

MA…