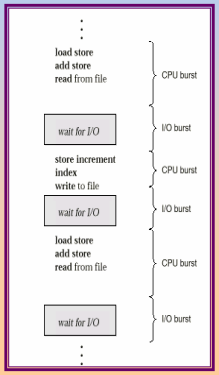
Scheduling della CPU

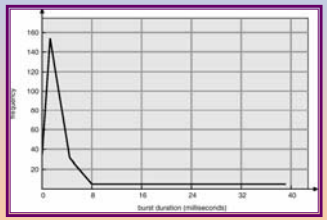
Scheduling della CPU

- Concetti fondamentali
- Criteri per lo scheduling
- Algoritmi di scheduling

Concetti fondamentali

- Sequenze (raffiche) di CPU (**CPU burst**) e di I/O (**I/O burst**)
  - L'esecuzione di un processo consiste di cicli di esecuzione da parte della CPU ed attese per operazioni di I/O
- La multiprogrammazione permette un maggiore sfruttamento della CPU





Distribuzione dei burst di CPU

Processi CPU-bound e I/O-bound

Scheduler della CPU (short-term)

- Sceglie un processo cui allocare la CPU, fra i processi (in memoria) pronti ad essere eseguiti.
- Lo scheduler può intervenire quando un processo:
  1. passa da stato **running** a stato **waiting**
     richiesta di I/O, o attesa di un evento (es., terminazione di un processo figlio)
  2. passa da **running** a **ready** (interrupt)
  3. passa da **waiting** a **ready** (completamento dell'I/O o dell'evento atteso)
  4. termina
- Due diversi schemi di scheduling:
  - ***Non–preemptive*** (senza prelazione), se lo scheduling è effettuato solo nei casi 1 e 4 (es. Windows) -> richiede tecniche di sincronizzazione
  - ***Preemptive***: la CPU può essere sottratta al processo attualmente running per essere assegnata ad un processo ready (casi 2 e 3)

Dispatcher

- Il modulo *dispatcher* passa il controllo della CPU al processo selezionato dallo scheduler a breve termine
- Il dispatcher effettua le seguenti operazioni:
  - Context switch
  - Passaggio a modo utente
  - Salto alla posizione corretta (program counter) del programma utente per riavviarne l'esecuzione
- *Latenza di dispatch*:
  - è il tempo impiegato dal dispatcher per sospendere un processo e avviare una nuova esecuzione.

Criteri di scheduling

- Misure considerate:
  - **Utilizzo della CPU**: percentuale di tempo in cui la CPU è usata per eseguire I processi utente
  - **Throughput**: numero di processi che completano la loro esecuzione nell'unità di tempo
  - **Tempo di turnaround**: tempo impiegato per eseguire un processo
  - **Tempo di attesa**: tempo speso dal processo nella ready queue
  - **Tempo di risposta**: tempo che intercorre tra la sottomissione di una richiesta e la prima risposta prodotta
- Criteri di ottimizzazione
  - Massimizzare l'utilizzo di CPU
  - Massimizzare il throughput
  - Minimizzare il tempo di turnaround (medio, o massimo)
  - Minimo **tempo di attesa** (medio , o massimo)
  - Minimo tempo di risposta (medio , o massimo, o varianza)

Scheduling First–Come–First–Served (FCFS)

ESEMPIO 1

Processo	Tempo di burst
P_1	24
P_2	3
P_3	3

- I processi arrivano al sistema nell'ordine: P_1 , P_2 , P_3.
- Il diagramma di Gantt per lo scheduling **FCFS** è:

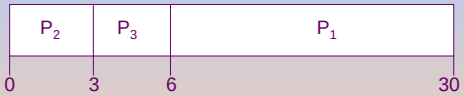
- Tempi di attesa: $P_1 = 0$; $P_2 = 24$; $P_3 = 27$.
- Tempo medio di attesa = (0 + 24 + 27)/3 = 17.

Scheduling FCFS

Se l'ordine di arrivo è

$$P_2 , P_3 , P_1,$$

il diagramma di Gantt risulta…



- Tempi di attesa: $P_1 = 6$; $P_2 = 0$; $P_3 = 3$.
- Tempo medio di attesa = (6 + 0 + 3)/3 = 3.
- Si può verificare l'*effetto convoglio*
  - processi I/O-bound devono attendere che un processo CPU-bound liberi la CPU
  - Scarso parallelismo fra CPU e canali di I/O

Scheduling Shortest–Job–First (SJF)

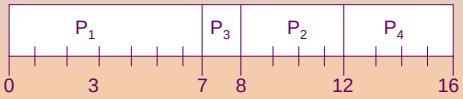
- Si opera lo scheduling in base alla brevità dei CPU burst.
  - Per ogni processo si tiene traccia della lunghezza del successivo burst di CPU
- Due schemi:
  - **non–preemptive** — una volta che la CPU è stata allocata al processo, non gli può essere prelazionata fino al termine del CPU burst corrente;
  - **preemptive** — la CPU può essere assegnata ad un nuovo processo con burst di CPU minore del tempo rimasto per il processo corrente
    - schema noto come ***Shortest–Remaining–Time–First*** (**SRTF**)
- **SJF** è *ottimale* rispetto al tempo di attesa
  - minimizza il tempo medio di attesa per un dato insieme di processi

Scheduling SJF non–preemptive

ESEMPIO 2

Processo	Tempo di arrivo	Tempo di burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- Diagramma di Gannt:



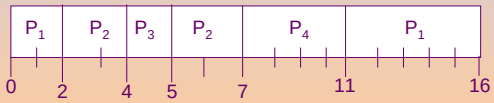
- Tempo medio di attesa = (0 + 6 + 3 + 7)/4 = 4.

Scheduling SJF preemptive (SRTF)

ESEMPIO 3

Processo	Tempo di arrivo	Tempo di burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- Diagramma di Gannt:



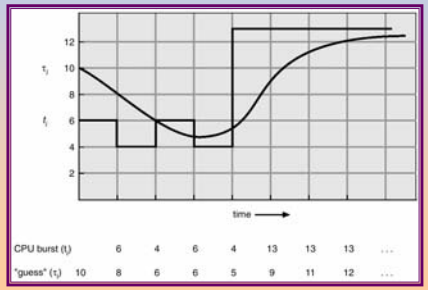
- Tempo medio di attesa = (9 + 1 + 0 +2)/4 = 3.

Predizione del CPU burst successivo

- La lunghezza del prossimo CPU burst può essere stimata in base alla lunghezza dei CPU burst precedenti, tramite una media esponenziale.

1. t_n = lunghezza dell' n – esimo CPU burst
2. τ_{n+1} = valore stimato del prossimo CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Si definisca :

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



Predizione del CPU burst: Esempi

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - La storia recente non è presa in considerazione.
- $\alpha = 1$
  - $\tau_{n+1} = t_n$
  - Viene considerato soltanto l'ultimo CPU burst.
- Espandendo la formula si ottiene:

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

- Poiché α e $(1-\alpha)$ sono entrambi minori o uguali ad 1, ciascun termine ha minor peso del suo predecessore.

Scheduling a priorità

- Un valore di priorità (intero) è associato a ciascun processo.
- La CPU viene allocata al processo con la priorità più alta
  - intero più basso ≡ priorità più alta
- Due schemi: preemptive / non–preemptive
- **SJF** è uno scheduling a priorità dove la priorità è rappresentata dal successivo tempo di burst.
- Problema = **Starvation** (blocco indefinito)
  - i processi a bassa priorità potrebbero non venir mai eseguiti.
  - Soluzione ≡ **Aging** (invecchiamento): aumento graduale della priorità dei processi che si trovano in attesa nel sistema da lungo tempo.

Scheduling Round Robin (RR)

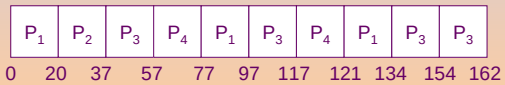
- Ogni processo ha diritto ad un intervallo di tempo di CPU: *quanto di tempo*
  - generalmente 10–100 millisecondi
- Passato un quanto di tempo, il processo è forzato a rilasciare la CPU e accodato alla ready queue
- Tempo di attesa controllato (dipende da quanto di tempo e nr. di processi)
  - Se ci sono n processi nella ready queue ed il quanto di tempo è q, ciascun processo occupa $1/n$ del tempo di CPU in frazioni di, al più, q unità di tempo
  - Nessun processo attende per più di $(n-1)\times q$ unità di tempo.
- Prestazioni:
  - q grande ⇒ FCFS
  - q piccolo ⇒ se è paragonabile al tempo di context switch, l'overhead è alto

Scheduling RR

ESEMPIO 4

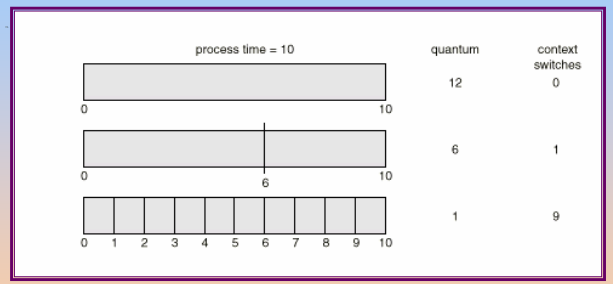
Processo	Tempo di burst
P_1	53
P_2	17
P_3	68
P_4	24

- Il diagramma di Gantt è (con quanto = 20):



- Prestazioni
  - In genere si ha un tempo medio di turnaround maggiore rispetto a **SJF**
  - tuttavia si ha una miglior tempo di risposta.

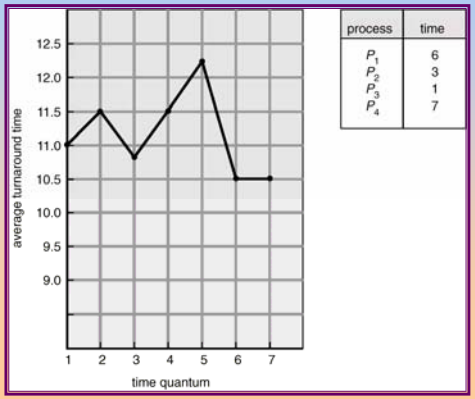
Scheduling RR -- Quanto di tempo vs. context switch



- Un quanto di tempo minore incrementa il numero di context switch
- Possono peggiorare le prestazioni
  - Utilizzo di CPU, throughput, tempi di turnaround,…

Scheduling RR -- Quanto di tempo e tempo di turnaround

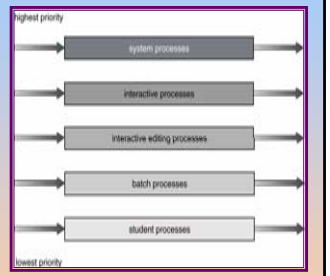
Variazione del tempo di turnaround in funzione della lunghezza del quanto di tempo



Empiricamente: il quanto di tempo deve essere maggiore dell'80% dei CPU burst.

Scheduling con code multiple

- La ready queue è suddivisa in code separate:
  - Esempio:
    - **foreground** (interattiva)
    - **background** (batch)
  - Ogni coda ha un algoritmo di scheduling:
    - foreground – **RR**
    - background – **FCFS**
- Bisogna stabilire lo scheduling tra code.
  - *Scheduling a priorità fissa*;
    - es. prima tutti i processi in foreground
    - Rischio di starvation!
  - *Time slice*:
    - Ogni coda ha un tempo di CPU che suddivide fra i propri processi.
    - Ad esempio:
      - 80% per foreground in **RR**
      - 20% per background in **FCFS**

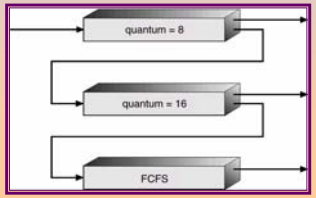


Code multiple con feedback

- Un processo può spostarsi fra le varie code
  - l'aging può essere implementato in questo modo.
- Lo scheduler con code multiple con feedback è definito dai seguenti parametri:
  - Numero di code
  - Algoritmi di scheduling per ciascuna coda
  - Metodo impiegato per determinare **quando** spostare un processo in una coda **a priorità maggiore**
  - Metodo impiegato per determinare **quando** spostare un processo in una coda **a priorità minore**
  - Metodo impiegato per determinare **in quale coda** deve essere posto un processo quando richiede un servizio

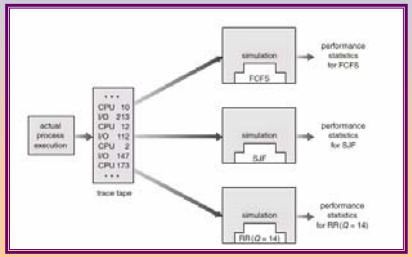
Esempio di code multiple con feedback

- Tre code:
  - Q_0 – **RR** con quanto di tempo di 8 millisecondi;
  - Q_1 – **RR** con quanto di tempo di 16 millisecondi;
  - Q_2 – **FCFS**.
- Scheduling:
  - Un nuovo processo P viene immesso nella coda Q_0.
  - Quando P prende possesso della CPU riceve 8 millisecondi.
  - Se P non termina, viene spostato nella coda Q_1.
  - Nella coda Q_1 P è ancora servito **RR** e riceve ulteriori 16 millisecondi.
  - Se P non ha ancora terminato, viene mosso nella coda Q_2.



Valutazione degli algoritmi

- Modellazione deterministica
  - prende in considerazione un carico di lavoro predeterminato e definisce le prestazioni di ciascun algoritmo per tale carico di lavoro.
- Modelli di code
  - modelli statistici del comportamento del sistema.
- Simulazioni.



Valutazione di scheduler di CPU tramite simulazione