Scheduling dei Processi

Sistemi Operativi

Antonino Staiano

Email: antonino.staiano@uniparthenope.it

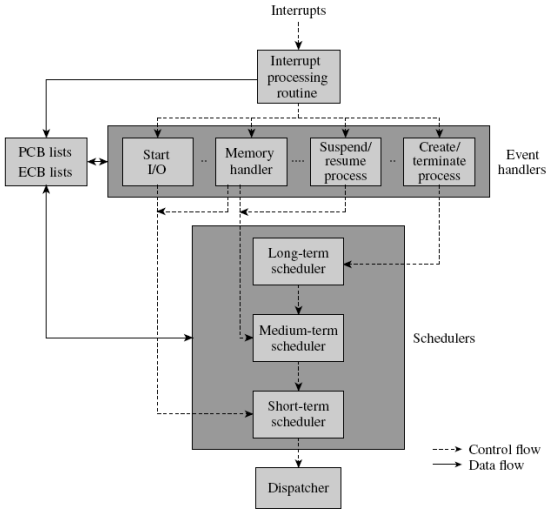
Scheduling in Pratica

- Per fornire una combinazione appropriata di prestazioni di sistema e servizio utente, il SO deve adattare il suo funzionamento alla natura e al numero delle richieste utente, e alla disponibilità delle risorse
  - Un singolo scheduler che usa una strategia di scheduling classica non può affrontare in modo efficace tutti questi problemi
- I SO moderni impiegano più scheduler
  - Fino a tre scheduler
- Alcuni scheduler possono usare una combinazione di diverse strategie di scheduling

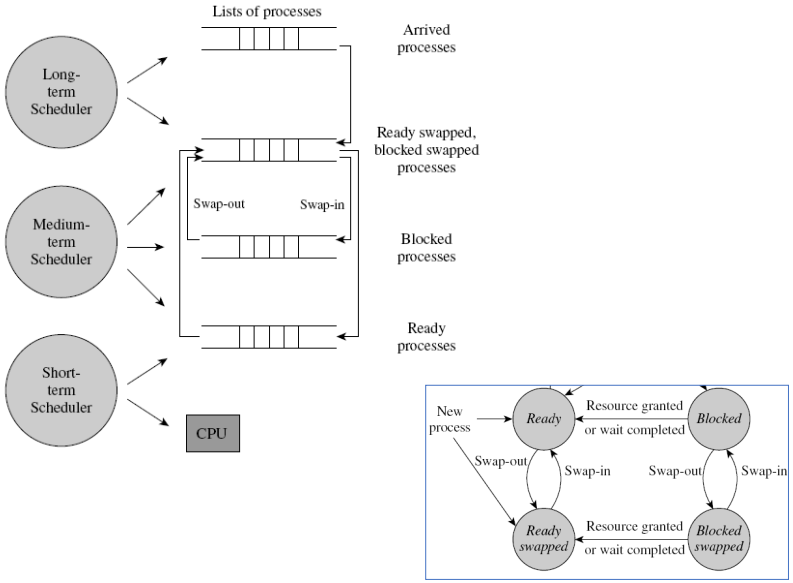
Scheduling in pratica: Scheduler a Lungo, Medio e Breve Termine

- Questi scheduler eseguono le funzioni seguenti:
  - **Lungo termine**: decide quando ammettere un processo appena arrivato allo scheduling, a seconda della:
    - Natura (se CPU-bound o I/O bound)
    - Disponibilità delle risorse
      - Strutture dati del kernel, spazio per lo swapping
  - **Medio termine**: decide quando fare swap-out di un processo dalla memoria e quando ricaricarlo in modo che ci sia un numero sufficiente di processi *ready* in memoria
  - **Breve termine**: decide quale prossimo processo ready servire con la CPU e per quanto tempo
    - Chiamato anche scheduler di processo, o scheduler

Gestione degli eventi e Scheduling

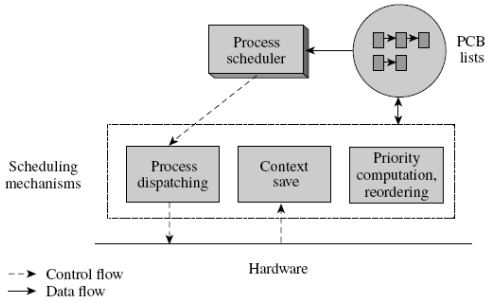


Esempio: Scheduling a Lungo, Medio e Breve Termine in un Sistema Time-Sharing



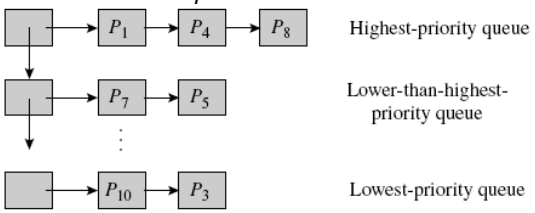
Scheduling in pratica: Strutture Dati e Meccanismi di Scheduling

- L'interrupt di routine di servizio invoca il salvataggio del contesto
- Il dispatcher carica due campi del PCB, PSW e GPR nella CPU per ripristinare le operazioni del processo selezionato
- Lo kernel esegue un *loop idle* se non ci sono processi ready



Scheduling in pratica: Scheduling basato su Priorità

- E' mantenuta una lista separata di processi ready per ogni valore di priorità
  - Organizzata come coda di PCB
- L'overhead dipende dal numero di priorità distinte, non dal numero dei processi ready
- Può portare a *starvation* dei processi a bassa priorità
  - L'aging può risolvere tale problema
- Può causare l'*inversione di priorità*
  -



Scheduling in pratica: Scheduling Round-Robin con Time Slicing

- Può essere implementato con una lista singola di PCB dei processi ready
  - La lista è organizzata come una coda
- Lo scheduler rimuove il primo PCB dalla coda e schedula il processo corrispondente
  - Se il *time slice* scade, il PCB è messo alla fine della coda
  - Se il processo avvia un'operazione di I/O, il suo PCB è aggiunto alla fine della coda quando la sua operazione di I/O è completata
- Il PCB di un processo ready si sposta verso la testa della coda fino a che il processo è schedulato

Scheduling in pratica: Scheduling Multilivello

- Combina lo scheduling con priorità e lo scheduling RR e gestisce diverse code di processi ready
- Ad ogni ready queue sono associati una priorità ed un time slice
  - All'interno viene eseguito uno scheduling RR con time slicing
  - La coda con alta priorità ha un piccolo time slice
    - Buoni tempi di risposta dei processi
  - La coda a basa priorità ha un time slice più grande
    - Basso overhead per la commutazione di processo
- Un processo in testa ad una coda è schedulato solo se le code per tutti i livelli più elevati di priorità sono vuote
- Lo scheduling è con prelazione
- Le priorità sono statiche

Scheduling in pratica: Scheduling Adattivo Multilivello

- Chiamato anche scheduling con feedback multilivello
- Lo scheduler varia la priorità di un processo in modo che esso abbia un time slice consistente con il suo requisito di CPU
- Lo scheduler determina il livello di priorità «corretto» per un processo osservando il suo uso recente di CPU e I/O
  - Sposta i processi a tale livello
- Esempio: CTSS, un SO time-sharing per l'IBM 7094 negli anni '60
  - Struttura di priorità ad otto livelli

Scheduling Real-Time

- Lo scheduling real-time deve gestire due vincoli di scheduling speciali e cercare, allo stesso tempo, di soddisfare le deadline delle applicazioni
  - Primo, i processi nelle applicazioni real-time sono processi interattivi
    - La deadline di un'applicazione dovrebe essere tradotta in deadline appropriate per i processi
  - Secondo, i processi possono essere periodici
    - Le differenti istanze di un processo possono arrivare a intervalli fissi e tutte che devono soddisfare le rispettive deadline

Precedenze di Processo e Schedulazioni ammissibili

- Sono considerate le dipendenze tra i processi (esempio, Pi $\longrightarrow$ Pj) mentre si determinano le deadline e lo scheduling

Un grafo delle precedenze dei processi *(PPG)* è un grafo diretto $G \equiv (N,E)$ tale che $P_i \in N$ rappresenti un processo, ed un arco $(P_i, P_j) \in E$ implica $P_i \rightarrow P_j$. Quindi, un cammino $P_i, \ldots, P_k$ in PPG implica $P_i \rightarrow^* P_k$. Un processo P_k è un discendente di P_i se $P_i \rightarrow^* P_k$.

- E' garantito che i requisiti di risposta siano soddisfatti (sistemi real-time hard) o soddisfatti probabilisticamente (sistemi real-time soft), a seconda del tipo di sistema RT
  - *Una **schedulazione ammissibile** è una sequenza di decisioni di scheduling che permette ai processi di un'applicazione di operare in accordo con le rispettive precedenze e di soddisfare i requisiti dell'applicazione*
- Data l'applicazione, lo scheduling RT è orientato sull'implementazione di una schedulazione ammissibile, se ne esiste una

Precedenze di Processo e Schedulazioni ammissibili (cont.)

Approcci allo scheduling Real-time

Approach	Description
Static scheduling	A schedule is prepared *before* operation of the real-time application begins. Process interactions, periodicities, resource constraints, and deadlines are considered in preparing the schedule.
Priority-based scheduling	The real-time application is analyzed to assign appropriate priorities to processes in it. Conventional priority-based scheduling is used during operation of the application.
Dynamic scheduling	Scheduling is performed when a request to create a process is made. Process creation succeeds only if response requirement of the process can be satisfied in a guaranteed manner.

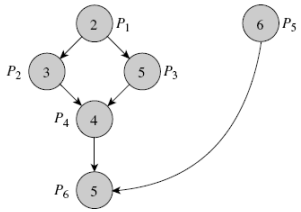
- Altra politica di scheduling dinamico: scheduling ottimista
  - Ammette tutti i processi; può perdersi qualche deadline

Scheduling con Deadline

- Possono essere specificati due tipi di deadline
  - Deadline di inizio: l'ultimo istante di tempo entro cui le operazioni del processo devono iniziare
  - Deadline di completamento: istante in cui le operazioni del processo devono terminare
- La stima della deadline è fatta considerando le precedenze dei processi e lavorando all'indietro dal requisito di risposta dell'applicazione

$$D_i = D_{application} - \sum_{k \in descendant(i)} x_k$$

Esempio: determinare le deadline dei processi



- Il totale dei tempi di servizio dei processi è 25 secondi
- Se l'applicazione deve produrre una risposta in 25 secondi, le deadline dei processi sarebbero:

Process	P_1	P_2	P_3	P_4	P_5	P_6
Deadline	8	16	16	20	20	25

Scheduling con Deadline (cont.)

- Determinare la deadline è più complesso in pratica
  - Deve considerare molti altri vincoli
    - Ad esempio, l'overlap delle operazioni di I/O con le elaborazioni della CPU
- Lo scheduling **Earliest Deadline First** (EDF) seleziona sempre il processo con la deadline più prossima
  - Sia **seq** la sequenza di in cui i processi sono elaborati
  - Se **pos(Pi)** è la posizione di Pi nella sequenza delle decisioni di scheduling, lo sforamento della deadline non avviene se

$$\sum_{k:pos(P_k) \leq pos(P_i)} x_k \leq D_i$$

  - La condizione vale quando esiste una schedulazione ammissibile
- Vantaggi: semplicità e natura senza prelazione
- Buona politica per lo scheduling statico

Scheduling con Deadline (cont.)

Time	Process completed	Deadline overrun	Processes in system	Process scheduled
0	–	0	$P_1 : 8, P_2 : 16, P_3 : 16, P_4 : 20, P_5 : 20, P_6 : 25$	P_1
2	P_1	0	$P_2 : 16, P_3 : 16, P_4 : 20, P_5 : 20, P_6 : 25$	P_2
5	P_2	0	$P_3 : 16, P_4 : 20, P_5 : 20, P_6 : 25$	P_3
10	P_3	0	$P_4 : 20, P_5 : 20, P_6 : 25$	P_4
14	P_4	0	$P_5 : 20, P_6 : 25$	P_5
20	P_5	0	$P_6 : 25$	P_6
25	P_2	0	–	–

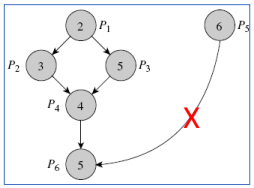
- P4: 20 indica che P4 ha la deadline a 20
- P2, P3 e P4, P5 hanno le stesse deadline
  - Sono possibili altre tre schedulazioni
  - Nessuna incorre nello sforamento delle deadline

Esempio: problemi con lo scheduling EDF

- Se rimuoviamo l'arco (P5, P6) dal PPG
  - Due applicazioni indipendenti: P1-P4 e P6, e P5
  - Se tutti i processi devono essere completati in 19 secondi
    - Non esiste una soluzione ammissibile
  - Deadline dei processi

Process	P_1	P_2	P_3	P_4	P_5	P_6
Deadline	2	10	10	14	19	19

  - Lo scheduling EDF può schedulare i processi come segue:
    - P1, P2, P3, P4, P5, P6 oppure P1, P2, P3, P4, P6, P5
    - Quindi il numero dei processi che non rispettano le rispettive deadline non è predicibile



Ammissibilità della schedulazione per Processi Periodici

- Frazione del tempo di CPU usato da Pi = xi / Ti
- Nell'esempio, le frazioni di tempo di CPU usato somma a 0.93

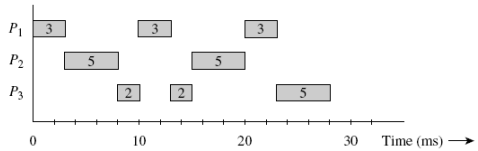
Process	P_1	P_2	P_3
Time period (ms)	10	15	30
Service time (ms)	3	5	9

  - Se l'overhead della CPU è trascurabile, è ammissibile servire i tre processi
- In generale, l'insieme dei processi periodici P1, …, Pn che non eseguono I/O può essere servito da un sistema hard RT che ha un overhead trascurabile se:

$$\Sigma_{i=1\ldots n} \frac{x_i}{T_i} \leq 1 \qquad (7.4)$$

Scheduling Rate Monotonic (SRM)

- Determina i tassi a cui il processo deve ripetersi
  - Tasso di Pi = 1 / Ti
- Assegna lo stesso rate come priorità del processo
  - Un processo con un periodo più piccolo ha maggiore priorità
- Impiega uno scheduling basato su priorità
  - Può completare le sue operazioni prima



Scheduling Rate Monotonic (SRM)

- SRM non garantisce uno scheduling ammissibile in tutte le situazioni
  - Per esempio, se P3 aveva un periodo di 27 secondi
- Se l'applicazione ha un elevato numero di processi, può non essere in grado di raggiungere più del 69% di utilizzo di CPU se deve rispettare le deadline dei processi

$$\sum_{i=1}^{m} \frac{xi}{Ti} \leq m\left(2^{\frac{1}{m}} - 1\right) \rightarrow 0.69 \; per \; m \rightarrow \infty$$

- L'algoritmo di *scheduling guidato da deadline* assegna dinamicamente le priorità ai processi sulla basa delle loro deadline correnti
  - Può raggiungere il 100% di utilizzo di CPU
  - Le prestazioni possono essere più basse a causa dell'overhead dell'assegnamento dinamico delle priorità