

## Politiche di ordinamento di processi *Indice*

- 1 Programmazione concorrente
- 2 Classificazione di sistemi
- 3 Politiche di ordinamento
  - 3.1 Per sistemi a lotti
  - 3.2 Per sistemi interattivi
  - 3.3 Per sistemi in tempo reale

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 31

## Programmazione concorrente - 1

- Molti sistemi sono inerentemente concorrenti
  - Occorrono strumenti per rappresentare e rendere esplicito il loro *parallelismo potenziale*
- **Programmazione concorrente** è l'insieme di notazioni e tecniche usate per
  - Esplicitare il parallelismo insito in un problema
    - Definendo e rappresentando singoli flussi di controllo come processi
  - Trattare i problemi di comunicazione e sincronizzazione da esso risultanti
- È disciplina distinta dalla realizzazione di parallelismo su uno o più elaboratori
  - Perché richiede la definizione di strategie e meccanismi di ordinamento dei processi (la seconda può non averne bisogno)

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 32

## Programmazione concorrente - 2

- Un programma concorrente corretto non richiede di specificare a priori l'esatto ordine di esecuzione dei suoi processi
  - Come in un buon sistema operativo moderno
- L'ordinamento locale (tra gruppi di processi correlati) può essere reso esplicitamente tramite primitive di comunicazione e sincronizzazione
- L'ordinamento globale non deve avere influenza sul risultato funzionale (i valori prodotti)

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 33

## Classificazione di sistemi - 1

- Diverse classi di sistemi concorrenti richiedono proprie politiche di ordinamento di processi
- 3 classi generali
  - Sistemi "a lotti" (*batch*)
    - Ordinamento predeterminato; lavori di lunga durata e limitata urgenza; prerilascio non necessario
  - Sistemi interattivi
    - Grande varietà di attività; prerilascio essenziale
  - Sistemi in tempo reale
    - Lavori di durata ridotta ma con elevata urgenza; l'ordinamento deve riflettere l'importanza del processo; prerilascio possibile

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 34

## Classificazione di sistemi - 2

- Caratteristiche desiderabili delle politiche di ordinamento
  - Per tutti i sistemi
    - **Equità** (*fairness*)
      - Nella distribuzione delle opportunità di esecuzione
    - **Coerenza** (*enforcement*)
      - Nell'applicazione della politica a tutti i processi
    - **Bilanciamento**
      - Nell'uso di tutte le risorse del sistema

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 35

## Classificazione di sistemi - 3

- Obiettivi di politiche per i **sistemi a lotti**
  - Massimo prodotto per unità di tempo (*throughput*)
  - Massima rapidità di servizio per singolo lavoro (*turn-around time*)
    - Media statistica
  - Massimo utilizzo delle risorse di elaborazione
- Obiettivi di politiche per i **sistemi interattivi**
  - Rapidità di risposta per singolo lavoro
  - Soddisfazione delle aspettative degli utenti
- Obiettivi di politiche per i **sistemi in tempo reale**
  - Rispetto delle scadenze (*deadline*)
  - Predicibilità di comportamento (*predictability*)

Politiche di ordinamento

Architettura degli elaboratori 2 - T. Vardanega

Pagina 36

### Politiche di ordinamento - 1

- Per sistemi a lotti
  - **FCFS** (*First come first served*)
    - Senza prerilascio, senza priorità
    - Ordine di esecuzione = ordine di arrivo
    - Massima semplicità, basso utilizzo delle risorse
  - **SJB** (*Shortest job first*)
    - Senza prerilascio, richiede conoscenza dei tempi richiesti di esecuzione
    - Esegue prima il lavoro (*job*) più breve
    - Non è equo con i lavori non presenti all'inizio
  - **SRTN** (*Shortest remaining time next*)
    - Aggiunge prerilascio a SJB
    - Esegue prima il processo più veloce a completare
    - Tiene conto di nuovi processo quando essi arrivano

Nota che parliamo di *lavori* quando operiamo senza prerilascio e di *processi* quando operiamo on prerilascio

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 37

### Politiche di ordinamento - 2

- Per sistemi interattivi (1/2)
  - **OQ** : Ordinamento a quanti (*round robin*)
    - Con prerilascio, senza priorità
    - Ogni processo esegue al più per un quanto alla volta
    - Lista circolare di processi
  - **OQP** : Ordinamento a quanti con priorità
    - Quanti diversi per livello di priorità
      - Come attribuire priorità a processi e come farle eventualmente variare
  - **GP** : Con garanzia per processo
    - Con prerilascio e con promessa di una data quantità di tempo di esecuzione (p.es.  $1/n$  per  $n$  processi concorrenti)
      - Le necessità di ciascun processo devono essere note (stimate) a priori
    - Esegue prima il lavoro maggiormente penalizzato rispetto alla promessa
      - Verifica periodica o ad evento (soddisfaccimento della promessa)

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 38

### Politiche di ordinamento - 3

- Per sistemi interattivi (2/2)
  - **SG**: Senza garanzia
    - Con prerilascio e priorità, opera sul principio della lotteria
      - Ogni processo riceve numeri da giocare
      - A priorità più alta corrispondono più numeri da giocare
    - Ad ogni scelta per assegnazione di risorsa, essa va al processo possessore del numero estratto
    - Le estrazioni avvengono periodicamente (= quanti) e/o ad eventi (p.es. attesa di risorse non disponibili)
  - Comportamento imprevedibile sul breve periodo, ma tende a stabilizzarsi statisticamente nel tempo
- **GU** : Con garanzia per utente
  - Come GP ma con garanzia riferita a ciascun utente (possessore di più processi)

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 39

### Politiche di ordinamento - 4

- Per sistemi in tempo reale (1/7)
  - I sistemi in tempo reale sono sistemi concorrenti nei quali il valore corretto deve essere prodotto entro un tempo fissato
    - Oltre tale limite, il valore prodotto ha utilità decrescente o nulla
  - L'ordinamento (*scheduling*) di processi deve fornire garanzie di completamento adeguate ai processi
    - Deve essere analizzabile staticamente (predicibile)
  - Il caso peggiore è sempre quando i processi sono tutti pronti per eseguire all'istante iniziale (*critical instant*)

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 40

### Politiche di ordinamento - 5

- Per sistemi in tempo reale (2/7)
  - Modello semplice (*cyclic executive*)
    - L'applicazione consiste di un insieme fissato di processi periodici (*ripetitivi*) ed *indipendenti* con caratteristiche note
    - Ciascun processo è suddiviso in una sequenza ordinata di procedure di durata massima nota
    - L'ordinamento è costruito a tavolino come una sequenza di chiamate a procedure di processi fino al loro completamento
    - Un ciclo detto maggiore (*major cycle*) racchiude l'invocazione di tutte le sequenze di tutti i processi
    - Il ciclo maggiore è suddiviso in N cicli minori (*minor cycle*) di durata fissa che racchiude l'invocazione di specifiche sottosequenze

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 41

### Esempio 1

#### Modello semplice senza suddivisione

Processo	Periodo T	Durata C
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

Conviene che i periodi siano armonici!

$$U = \sum_i (C_i / T_i) = 46/50 = 0.92$$

Ciclo maggiore di durata 100 → MCM di tutti i periodi  
Ciclo minore di durata 25 → periodo più breve

Politiche di ordinamento
Architettura degli elaboratori 2 - T. Vardanega
Pagina 42

### Politiche di ordinamento - 6

- Per sistemi in tempo reale (3/7)
  - Ordinamento a priorità fissa
    - Preferibilmente *con* preilascio (a priorità!)
      - Processi periodici, indipendenti e noti
    - Assegnazione di priorità secondo il periodo (*rate monotonic*)
      - Per scadenza uguale a periodo ( $D = T$ ), priorità maggiore per periodo più breve
      - Test di ammissibilità sufficiente ma non necessario per  $n$  processi (Liu e Layland, 1973)

$$U = \sum_{i=1}^n \left( \frac{C_i}{T_i} \right) \leq f(n) = n(2^{1/n} - 1)$$

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 43

### Esempio 2

#### Caso semplice ordinamento a priorità

Processo	Periodo T	Durata C	Priorità
A	80	40	1 ←
B	40	10	2
C	20	5	3 ←

← Bassa  
← Alta

Il test di ammissibilità fallisce  $U = 1 > f(3) = 0,78$  ma il sistema è ammissibile!

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 44

### Politiche di ordinamento - 7

- Per sistemi in tempo reale (4/7)
  - Ordinamento a priorità fissa con preilascio e scadenza inferiore a periodo ( $D < T$ )
    - Assegnazione di priorità secondo la scadenza
    - Modello di processi generalizzato
      - Condivisione di risorse e processi sporadici
    - Rischio di **inversione di priorità**
      - Processi a priorità maggiore *bloccati* dall'esecuzione di processi a priorità minore
      - Effetto causato dall'accesso esclusivo a risorse condivise
      - Può condurre a blocco circolare (*deadlock*)

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 45

### Politiche di ordinamento - 8

- Per sistemi in tempo reale (5/7)
  - Innalzamento della priorità (*priority ceiling*) - 1
    - Ogni processo  $j$  ha una priorità statica di base fissata  $P_B$
    - Ogni risorsa condivisa  $i$  ha una priorità (*ceiling*)  $PC_i$  pari alla massima priorità dei processi che la stanno richiedendo
    - Ogni processo  $j$  ha una priorità dinamica  $P_j = \max\{P_B, PC_i\}$   $\forall$  risorsa  $i$  in suo possesso che causa blocco
    - Un processo può acquisire una risorsa solo se la sua priorità dinamica corrente è maggiore del *ceiling* di tutte le risorse attualmente in possesso di altri processi
      - Un processo a priorità maggiore può essere bloccato *una sola volta* durante l'intera sua esecuzione
      - Non vi è più rischio di *deadlock*

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 46

### Politiche di ordinamento - 9

- Innalzamento delle priorità (*ceiling*) - 2
  - Versione originale (*Basic Priority Inheritance*)
    - L'innalzamento avviene solo quando un processo a priorità maggiore si blocca all'ingresso di una risorsa in possesso di un processo a priorità inferiore
    - Richiede il controllo delle condizioni di blocco
  - Versione semplificata (*Immediate Ceiling Priority*)
    - $PC_i$  diventa il massimo tra le priorità dei processi che possono usare la risorsa
    - $P_j = \max\{P_B, PC_i\} \forall$  risorsa  $i$  in suo possesso
    - L'innalzamento avviene non appena il processo acquisisce la risorsa (prima ancora che il blocco possa avvenire)
    - Un processo può subire blocco solo prima di acquisire effettivamente la CPU

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 47

### Politiche di ordinamento - 10

- Per sistemi in tempo reale (7/7)
  - Calcolo del tempo di risposta  $R_i$  del processo  $i$ 
    - Tempo di **blocco** di processo  $i$ 
      - $B_i = \max_k \{C_k\} \forall$  risorsa  $k$  usata da processi a priorità più bassa di  $i$
    - **Interferenza** subita da  $i$  da parte di tutti i processi  $j$  a più alta priorità
      - $I_i = \sum_j \lceil R_j / T_j \rceil C_j$
    - $R_i = C_i + B_i + I_i$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_j}{T_j} \right\rceil C_j$$

$$a_i^{k+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{a_j^k}{T_j} \right\rceil C_j$$

$$a_i^0 = C_i$$

Politiche di ordinamento Architettura degli elaboratori 2 - T. Vardanega Pagina 48