

Lo scheduling

Un processo durante la sua evoluzione è o running o in attesa di un evento. Nel secondo caso trattasi della disponibilità di una risorsa (CPU , I/O, struttura dati, ecc.) di cui il processo attende l'allocazione.

Se un processo è in attesa di un evento il suo PCB è in una coda da cui sarà rimosso quando l'evento si verifica.

Queste code sono dette code di scheduling ed è detto schedulatore o scheduler il segmento del S.O. che effettua la selezione del processo in coda a cui allocare la risorsa e algoritmo di scheduling l'algoritmo che esso implementa

Tipici schedulatori

- lo schedulatore a breve termine o scheduler della CPU
- lo schedulatore a medio termine o schedulatore di swap (swapper)
- lo schedulatore a lungo termine o schedulatore di job
- lo schedulatore del disco, della stampante, ecc.

Obiettivi di un algoritmo di scheduling

Un algoritmo di scheduling persegue uno o più obiettivi quali:

- imparzialità: ad es. garantire ad ogni processo la sua giusta quota di tempo di CPU
- efficienza: ad es. tenere la CPU impegnata al 100% del tempo
- tempo di risposta: ad es. minimizzare il tempo di risposta per gli utenti interattivi
- turnaroud: minimizzare il tempo che gli utenti batch devono attendere per ricevere i risultati dei propri job
- throughput: massimizzare il numero di job elaborati per ora
- tempo medio di attesa: minimizzare il tempo medio che un processo deve attendere per accedere ad una risorsa

Il prerilascio

Gli algoritmi di scheduling si suddividono in:

- algoritmi a **prerilascio** (preemptive)
- algoritmi a **completamento** (non preemptive)

I primi, a differenza dei secondi, ammettono che il S.O. possa forzare il rilascio di una risorsa da parte di un processo e, quindi, attivare lo scheduling per la sua nuova assegnazione.

Il rilascio forzato implica necessariamente che sia possibile salvare lo stato di avanzamento del processo sulla risorsa in modo da permettere successivamente la sua ripresa.

Scheduling first come - first served (FCFS)

- tipico algoritmo di scheduling non preemptive
- ha il vantaggio della semplicità
- può dar luogo a tempi medi di attesa in coda molto lunghi.
- consente la gestione efficiente di una risorsa
- può essere indicato per lo scheduling di processi in attesa di risorse usualmente disponibili
- non può essere utilizzato come algoritmo di scheduling della CPU in sistemi time sharing
- nello scheduling della CPU può dar luogo all'effetto convoy.

Scheduling Round Robin (RR)

Tipico algoritmo di scheduling della CPU per sistemi time_sharing

Nasce dalla trasformazione preemptive del FCFS mediante l'impiego di un timer

Le prestazioni dell'algoritmo sono dipendenti dalla durata del periodo del timer che definisce il **quanto di tempo** di massimo utilizzo della risorsa o **time slice**

Aumentando il time slice tende a trasformarsi in FCFS mentre diminuendolo aumentano i salvataggi e ripristini di stato dei processi e quindi l'overhead del S.O..

Scheduling a priorità

Si parla di scheduling a priorità quando la scelta del processo tra i processi in attesa è fatta in funzione di una priorità che può essere sia statica che dinamica.

Per impedire che processi prioritari facciano uso indefinito di una risorsa (starvation uno scheduling a priorità è sempre preemptive

Es. di priorità statica: la priorità di un processo è definita a priori in funzione della sua importanza

Es di priorità dinamica: la priorità di un processo dipende dalla frazione di tempo, rapportata alla slice, di ultimo utilizzo della risorsa per cui attende

Scheduling a code multiple

Algoritmi di scheduling a priorità che utilizzano per i processi tante code di attesa quanti sono i livelli di priorità previsti.

Ad ogni coda è associato un quanto di tempo di utilizzo massimo della risorsa gestita decrescente al crescere della priorità

La priorità di un processo varia dinamicamente. Tipicamente aumenta se è piccola la frazione di slice spesa per utilizzare in precedenza la risorsa e diminuisce se è stata utilizzata l'intera slice. La priorità può anche aumentare con l'aumentare dell'attesa in coda (aging)

Scheduling shortest job (process) first SJF

Nasce come algoritmo di scheduling a lungo termine

E' l'algoritmo ottimale per minimizzare il tempo medio di attesa di assegnazione di una risorsa

Richiede di conoscere a priori per quanto tempo una risorsa sarà impiegata dal processo una volta che la ha ottenuta. Poiché questo di solito non è noto si basa su una sua stima.

Nel caso dello scheduler a lungo termine il tempo è quello stimato dall'utente.

Nel caso di scheduling di un processo può essere stimato con una media esponenziale che tiene conto dei tempi di utilizzo precedenti

Scheduling shortest job (process) first SJF

Media esponenziale:

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

- τ_{n+1} , τ_n tempi stimati
- t_n tempo effettivo
- α coefficiente $0 \leq \alpha \leq 1$

Algoritmo pesante ma semplice da implementare per $\alpha = 1/2$

Esempio: lo scheduling della CPU in UNIX

Lo scheduling si basa su una struttura di code a più livelli di priorità

La priorità di un processo in stato utente varia nel tempo mentre quella in stato kernel viene ridefinita in funzione dell'evento quando il processo va in attesa .

Il ricalcolo della priorità in stato utente tende a privilegiare i processi che in tempi recenti hanno effettuato CPU burst brevi.

Modalità di ricalcolo: Ogni secondo un daemon divide per due il contatore di utlizzi recente della CPU il cui valore definisce la nuova priorità e quindi la nuova posizione dei processi nelle code. Il contatore di utilizzo viene incrementato di 1 ogni 20 msec quando il processo è running.

Scheduling garantito

Divide il tempo di utilizzo di una risorsa tra gli n processi presenti

La priorità di un processo in attesa dipende dall'utilizzo complessivo precedente della risorse rapportato a $1/n$

Scheduling a lotteria

Ad un processo sono assegnati uno o più ticket. Lo scheduler estrae un ticket per selezionare il processo in attesa

Lo scheduling nei sistemi Real Time

E' essenzialmente lo scheduling della CPU.

Lo scheduler deve garantire che al manifestarsi di un evento un processo termini entro un tempo massimo predefinito.

L'evento può essere sia periodico che aperiodico

Affinché un sistema possa gestire eventi multipli periodici deve essere schedabile ossia:

$$\sum_{i=1}^m C_i/P_i \leq 1$$

dove m sono gli eventi, P_i il periodo dell'evento i-esimo e C_i i secondi di CPU richiesti per gestirlo

Algoritmi di scheduling real time per eventi periodici

Algoritmo rate monotonic

Ciascun processo ha una priorità pari alla frequenza di attivazione e lo scheduler assegna sempre la CPU al più prioritario applicando il preilascio se necessario.

Algoritmo earliest deadline first

Il scheduler sceglie il processo la cui successiva attivazione (deadline) è la più prossima nel tempo.

Algoritmo laxity

Lo scheduler sceglie il processo la cui elaborazione termina più vicina al suo tempo limite massimo per terminare.