Round Robin

martedì 11 aprile 2023 12:37

Scheduling Round-Robin (Time-Slicing)

Consente a tutti i processi di ottenere il controllo della CPU ed evita quindi il problema dell'attesa indefinita (starvation).

L'algoritmo di scheduling RR (round-robin) è un particolare algoritmo con prelazione (preemptive) che esegue i processi nell'ordine d'arrivo, come il FCFS, ma esegue la prelazione del processo in esecuzione, ponendolo alla fine della coda dei processi in attesa, qualora l'esecuzione duri più della "quantità di tempo" stabilita, e facendo proseguire l'esecuzione al successivo processo in attesa.

Ad esempio nell'ipotesi che vi siano i seguenti processi in coda con relativa durata in millisecondi, e la quantità di tempo stabilita di 20 ms:

- **p1: 30**
- **p2: 15**
- **p3: 60**
- **p4: 45**

Verranno eseguiti nel seguente ordine:

- **p1 (interrotto dopo 20 ms, ne rimangono altri 10)**
- **p2 (termina la propria esecuzione perché dura meno di 20 ms)**
- **p3 (interrotto dopo 20 ms, ne rimangono altri 40)**
- **p4 (interrotto dopo 20 ms, ne rimangono altri 25)**
- **p1 (termina la propria esecuzione perché necessitava di meno di 20 ms)**
- **p3 (interrotto dopo 20 ms, ne rimangono altri 20)**
- **p4 (interrotto dopo 20 ms, ne rimangono altri 5)**
- **p3 (termina la propria esecuzione perché necessitava di esattamente 20 ms)**
- **p4 (termina la propria esecuzione)**

i processi nello stato Ready vengono mandati in esecuzione a turno per uno specifico **quanto di tempo**

vi è prelazione, quindi un processo può essere bloccato anche se non ha completato la sua traccia o non vuole rilasciare la CPU spontaneamente

Svantaggi

- sfavorisce processi I/O bound rispetto a processi CPU bound
- non propriamente adeguato per la gestione di processi interattivi
- può causare sottoutilizzo dei dispositivi di I/O a causa del fatto che i processi I/O bound vengono sfavoriti

Si definiscono CPU bound in informatica i processi che sfruttano pesantemente le risorse computazionali del processore, ma non richiedono servizi di ingresso/uscita dati al sistema operativo in quantità rilevanti.

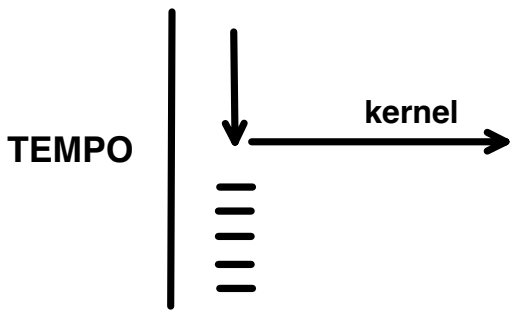
Esempio: Programmi di calcolo.

In informatica, I/O bound si riferisce alla condizione in cui il tempo necessario a compiere un'elaborazione è determinato principalmente dall'attesa delle operazioni di input/output. Ciò si contrappone ai processi CPU bound. Questa circostanza si verifica quando i dati vengono richiesti ad una velocità minore di quella a cui sono "consumati", o, in altre parole, si spende più tempo a richiedere i dati che ad elaborarli[1].

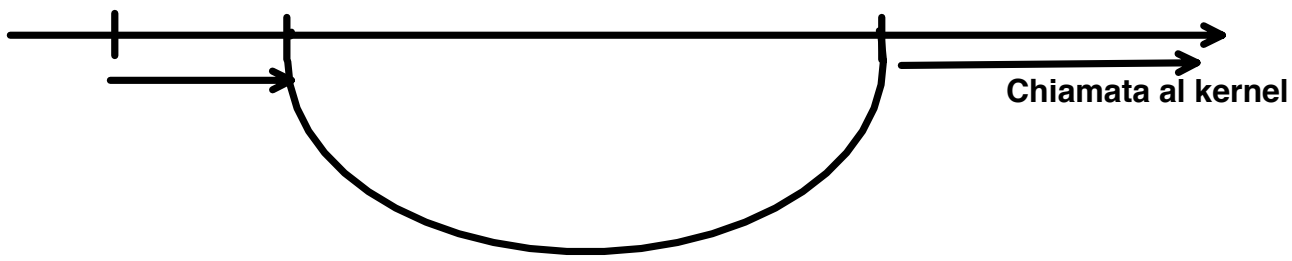
È inoltre da tenere in considerazione l'impatto dovuto ai frequenti context switch effettuati. È necessario quindi calcolare correttamente la durata ottimale della quantità di tempo per far sì che l'incidenza dei cambi di contesto sia abbastanza limitata, come anche i tempi di attesa. Si può stabilire che, per un funzionamento ottimale, le sequenze di operazioni di CPU dovrebbero essere più brevi della quantità di tempo stabilita in circa l'80% dei casi.

La commutazione di contesto (in inglese context switch), in informatica, indica una particolare operazione del sistema operativo che conserva lo stato del processo o thread, in modo da poter essere ripreso in un altro momento. Questa attività permette a più processi di condividere la CPU, ed è anche una caratteristica essenziale per i sistemi operativi multitasking.
Un altro significato del context switch è quello dovuto agli interrupt, ovvero quando un processo richiede l'uso di un disco o di attività di I/O, si richiede alla CPU di fermare il processo in esecuzione, salvare il suo stato con tutte le informazioni dei registri e altre varie informazioni, e di occupare la CPU nel soddisfare la richiesta di interrupt.

Noi assegniamo l'utilizzo della CPU in un lasso di tempo ad un processo I/O BOUND, ma questo processo non è in grado di usarla tutta.
Un processo I/O BOUND esegue con un po' di CPU e poi chiama un servizio del Kernel per andare a lavorare con qualche dispositivo di I/O. Quando il dispositivo di I/O viene attivato dal software del sistema che prende il controllo, noi rimaniamo in stato di blocco! Verremo risvegliati più avanti e verremo riportati in cpu più avanti se la CPU è libera, altrimenti sicuramente persisteremo nello stato ready per un po' di tempo. Il quanto di tempo può essere di durata superiore rispetto al tempo che serviva al processo I/O bound per lavorare in CPU e poi chiamare l'interazione col device.
Di questo tempo il processo I/O bound utilizza solo una frazione e l'altra frazione non la utilizziamo perché il software del sistema riprende il controllo e dopo aver attivato il dispositivo di I/O con cui vogliamo andare a colloquiare, assegnerà il controllo ad un altro processo.



Il problema della gestione dei processi interattivi è dovuto dal fatto che, se noi abbiamo un processo P che deve attivare un device, e il quanto di tempo scade prima che questo processo attivi un device, a quel punto rimane per un tot di tempo nello stato ready, quando invece dopo poche istruzioni macchina avrebbe potuto far sì che anche un dispositivo di I/O sarebbe stato attivato.



L'applicazione interattiva voleva usare la CPU per una quantità di istruzioni macchina iniziale prima di chiamare il kernel, ma io ho assegnato un quanto di tempo talmente piccolo che l'applicazione, non appena il quanto di tempo è scaduto, non è riuscita a completare il lavoro prima di chiamare il kernel per andare in I/O. Quindi quando quel quanto di tempo finisce abbiamo un interrupt, il kernel riprende il controllo e assegna la CPU a qualcun altro, l'applicazione che è stata interrotta dovrà in futuro riprende il controllo al punto esatto in cui è stata interrotta ma questo accadrà quando nel TIME-SHARING altre applicazioni che sono ready avranno altresì utilizzato la CPU.
Abbiamo un ritardo nell'attivazione di un dispositivo di I/O in base al ritardo di questa chiamata al kernel per attivare il dispositivo. Questa è una situazione problematica.
Per cercare di ovviare a questi problemi ci sono stati degli avanzamenti.