Algoritmi di scheduling

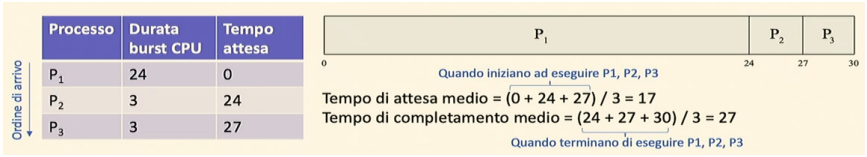
Leonardo Bizzoni

February 6, 2024

1 Scheduling per carichi prevalentemente batch

1.1 Scheduling first come first served (FCFS)

Algoritmo di scheduling non-preemptive. La CPU viene assegnata al primo processo, in stato *ready*, che la richiede.



1.1.1 Vantaggi

- Semplice da implementare, è una queue.
- Buon utilizzo della CPU, non c'è un'istante di tempo dove la CPU è idle.

1.1.2 Svantaggi

- Tempo di attesa medio può essere lungo

1.2 Scheduling per brevità - Shortest Job First (SJF)

Algoritmo di scheduling non-preemptive. La CPU viene assegnata al processo che ha il CPU burst più breve.

Ordine di arrivo

Processo	Durata burst CPU	Tempo attesa
P_1	6	0
P_2	8	6
P_3	7	14
P_4	3	21

Ordine per brevità

Processo	Durata burst CPU	Tempo attesa
P_4	3	0
P_1	6	3
P_3	7	9
P_2	8	16

Tempo di attesa medio SJF = (0 + 3 + 9 + 16) / 4 = 7
Tempo di completamento medio SJF = (3 + 9 + 16 + 24) / 4 = 13

1.2.1 Vantaggi

- Semplice da implementare, è un heap.
- Minimizza il tempo di attesa medio.

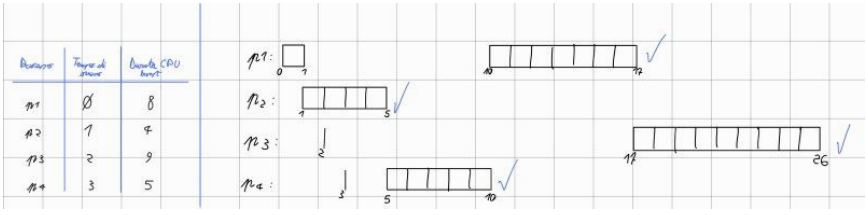
1.2.2 Svantaggi

- Non è sempre noto in anticipo quale processo avrà il CPU burst più breve.

1.3 Scheduling per tempo rimanente - Shortest Remaining Time First (SRTF)

Algoritmo di scheduling preemptive. La CPU viene assegnata al processo più vicino al completamento.

Se un processo in esecuzione arriverà al completamento in più tempo rispetto ad un processo ready, questo viene interrotto forzatamente.



Tempo di attesa = Istante terminazione−(Tempo arrivo+Durata CPU burst)

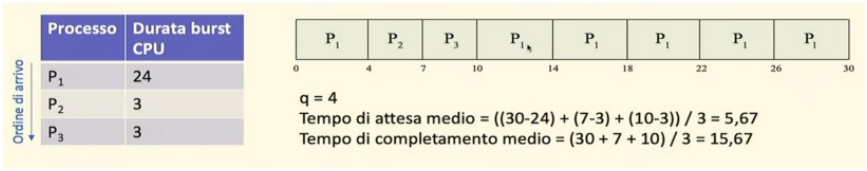
Tempo di completamento = Istante terminazione−Tempo arrivo

2 Scheduling per carichi prevalentemente interattivi

2.1 Scheduling circolare - Round Robin (RR)

Algoritmo di scheduling preemptive. Ogni processo ottiene una piccola quantità fissata di tempo di CPU (**quanto di tempo**) per il quale può esseere in esecuzione. Terminato il quanto di tempo, il processo viene **forzatamente** interrotto ed incodato nella ready queue.

In questo modo la ready queue funziona essenzialmente come un **buffer cicolare** ed i processi vengono scanditi dal primo all'ultimo, per poi ripartire dal primo nello stesso ordine.



2.1.1 NB

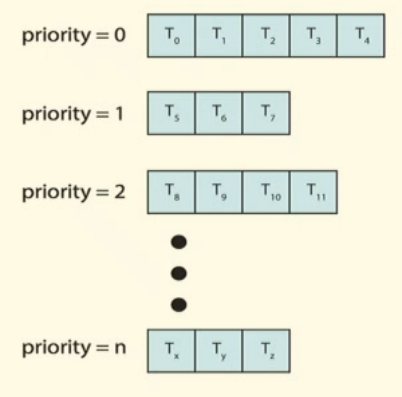
- Con un quanto di tempo troppo lungo, l'algoritmo è più simile ad un FCFS.
- Con un quanto di tempo troppo breve, diventa rilevante la latenza di dispatch e quindi la CPU è occupata al 50% per fare il dispatch dei processi ed al 50% per esecuire il processo.

3 Scheduling per carichi prevalentemente real-time (*tempo di risposta ad un evento minimo*)

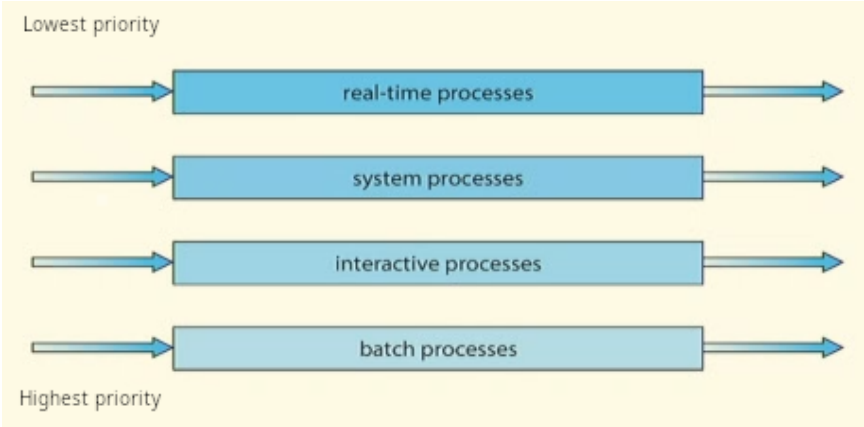
3.1 Scheduling con priorità

Può essere implementato come un algoritmo di scheduling sia preemptive che non-preemptive. Ad ogni processo viene associata una **priorità** e viene eseguito il processo con priorità minima (*In Windows è il contrario*).

A parità di priorità deve essere utilizzato un altro algoritmo di scheduling per stabilire quale processo eseguire, tipicamente il round-robin. Vengono utilizzate code diverse per priorità diverse. Nel momento in cui la coda dei processi con priorità N è vuota lo scheduler passa alla coda dei processi con priorità $N + 1$.



Nei sistemi operativi moderni il criterio usato per assegnare la priorità ad un processo è in base al tipo. Processi real-time o cyber-fisici hanno priorità minima mentre processi batch che effettuano lunghe computazione hanno tipicamente priorità elevate. Il riconoscimento del tipo viene effettuato dinamicamente durante l'esecuzione del processo, se questo ha lunghi CPU burst allora la sua priorità viene alzata, se invece ha molti I/O burst allora la sua priorità viene abbassata (*inizialmente un processo ha priorità minima*).



3.1.1 Problema dell'attesa infinita

Un processo con priorità troppo alta potrebbe restare in attesa per sempre(**starvation**). Per risolvere questo problema si diminuisce la priorità di un processo al crescere del tempo di permanenza nella ready queue(**aging/invecchiamento**).