

SSTF SCHEDULING > S.3.2



L'algoritmo di scheduling Shortest

Sob First prevede di impostare

una priorità facendo eseguire prima

i processi con prossimo CPU burst (e non la lunghezza totale)

Se 2 processi hanno la stessa durata del prossimo burst, allora si usa la FCFS.

- Vantaggi: L'average waiting time è ottimale
- Svantaggi: Necessita di un meccanismo di previsione dei next burst dei processi.

MECANISMO DI PREVISIONE

Per prevedere il next CPU burst, si ipotizza che sia simile al precedente. Per questo si usa la media esponenziale:

$$\overline{T}_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \overline{T}_n$$

• α è un "peso" usato per far valere di più la storia recente o passata.

Dove:

- t_n : è la misura del burst "attuale" o storia recente
- \overline{T}_n : sono i burst passati

Dalla formula si possono identificare 3 casi:

$$\bullet \alpha = 0 : \overline{T}_{n+1} = \overline{T}_n$$

Ovvero la storia recente non vale.

$$\bullet \alpha = 1 : \overline{T}_{n+1} = t_n$$

Ovvero la storia passata non vale.

$$\bullet 0 < \alpha < 1 : \overline{T}_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \alpha \cdot t_{n-1} + \dots + (1-\alpha)^{n+1} \cdot t_0$$

Ovvero la storia meno recente ha meno valore.

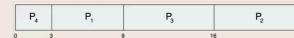
DIFF. TRA CON O SENZA PRELAZIONE

SSTF può essere con o senza prelazione:

Example of SJF

Processes	Burst time
P ₁	6
P ₂	8
P ₃	7
P ₄	11

• SJF scheduling chart



$$\text{Average waiting time} = (3 + 16 + 9 + 0) / 4 = 7$$

Quando si assegna la CPU, non si interrompe il processo fino a quando esso stesso non decide di andare in wait o terminare.

• preemptive: (Shortest Remaining Time First)

Se, durante l'esecuzione del processo, arriva un nuovo processo nella ready queue con burst minore rispetto al burst rimanente del proc. in esecuzione, si interrompe il corrente e si avvia quello nuovo.

Nell'esempio:

- P₁: Parte da 10 ma viene eseguito 1 => (10-1)

- P₂: Parte da 1 ma appare da 1 (arrival) => 1-1

e così via.

PRIORITY SCHEDULING > 5.3.4

Definisce un campo di priorità come N da assegnare.

La priorità può essere bassa o alta a seconda del numero (in alcuni sistemi 0 = max priorità in altri bassa).

SALVAGGIO CARPO

Il campo viene salvato nel PCB. Per poter determinare quale sarà il prossimo da eseguire, bisogna usare strutture dati adeguate.

DEFURZIONE E PREEMPTION

La priorità può essere definita

per: Lunghezza CPU burst

• Internamente: Misurate secondo criteri interni al S.O.

• Esternamente: Misurate secondo criteri esterni

(es. importanza task per l'utente)

E possono essere

• Non-preemptive:

Aspetta che il processo vada in wait o termine. All'arrivo di

Example of Shortest-remaining-time First

- Now we add the concepts of varying arrival times and preemption to the analysis



Preemptive SJF Gantt Chart

Processes	Arrival time	Burst time
P ₁	0	8
P ₂	1	4
P ₃	2	9
P ₄	3	5

$$\text{Average waiting time} = [(10-1)+(1-1)+(17-2)+5-3]/4 = 26/4 = 6.5 \text{ msec}$$

un nuovo proc. con priorità minore lo mette in coda, mantenendo la coda ordinata.

• Pre-emptive:

Effettua dei controlli periodici sulla ready queue.

Se un processo ha priorità maggiore di quello in esec. si interrompe l'ultimo e si esegue il proc. nuovo.

STARUATION

Indica i processi con priorità bassa che non vengono mai serviti perché arrivano sempre proc con priorità magg.

Questo problema si può risolvere con:

• Aging

Si fa aumentare le priorità dei processi man mano che passa il tempo.

• Combo con Round Robin

Si eseguono processi con la stessa priorità in round robin.

ROUND ROBIN SCHEDULING ➤ S.3.3

È una variante del FCFS che introduce la prelazione.

Si basa su una quantità di tempo (quant / time slice) in cui il processo verrà eseguito prima di cominciare.

Ci saranno, quindi, 2 casi:

- Tl proc. dura più del quant:

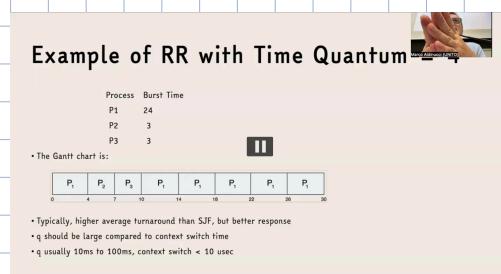
Il processo viene eseguito esattamente per il quinto e poi comincia.

- Il proc. dura meno del quatt:

Viene eseguito per la sua qualità e
volontariamente cede il controllo.

PRESTAZIONI

Il round robin ha un buon response time ma generalmente impiega un tempo medio maggiore (turnaround time)



La dimensione del quatt può influire sulle performance

- se troppo alto, si ricade nel FCFS
- se troppo basso, si rischia di aggiungere latenza di dispatch inutile.

MULTI LEVEL SCHEDULING ➤

È un tipo di scheduling che usa N code, ognuna potenzialmente col proprio algoritmo di scheduling.

Tipi di code

Si possono definire come

- code con priorità fissa

Vengono prima eseguiti i proc. della coda con priorità alta e poi a scendere.

- code in base al tipo di processo

Diversi tipi di processo hanno esigenze di scheduling differenti.

- Porzioni di tempo

Si assegnano delle percentuali di tempo per ogni coda.

Siccome nelle code con priorità si può avere starvation, sarà quindi necessario avere policy di starvation e round robin.

MULTI LEVEL FEEDBACK QUEUE

È un tipo di scheduling multi-levello in cui si definiscono code con bassa e alta priorità e si spostano i processi in base all'uso della CPU.

Se sfiora la durata della coda viene spostato più in basso.

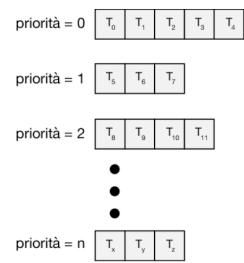


Figura 5.7 Code distinte per ogni priorità.

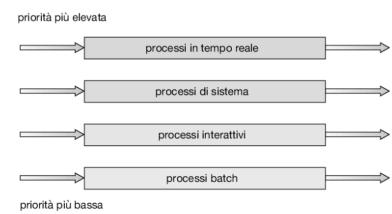


Figura 5.8 Scheduling a code multilivello.

Example of Multilevel Feedback Queue

• Three queues:

- Q0 - RR with time quantum 8 milliseconds
- Q1 - RR time quantum 16 milliseconds
- Q2 - FCFS

• Scheduling

- A new job enters queue Q0 which is served FCFS
 - When it gains CPU, job receives 8 milliseconds
 - If it does not finish in 8 milliseconds, job is moved to queue Q1
 - At Q1 job is again served FCFS and receives 16 additional milliseconds
 - If it still does not complete, it is preempted and moved to queue Q2

