Tipi di algoritmi di scheduling

Scheduling nei sistemi batch

Abbiamo tre tipi di algoritmi nei sistemi batch:

- **First come first server**: senza prelazione, singola coda di processi e i processi sono assegnati nell'ordine in cui arraiva alla coda. Molto facile da implementare e ci sta un principio d'equità, le prestazioni non sono ottimali in scenari di processi misti.
- **Shortest Job First**: Faccio avanzare i compiti più brevi, il quanto di **tempo deve essere noto a priori** altrimenti non ci sarebbe criterio di scelta. riesce a minimizzare il tempo medio per eseguire i processi. Ottimale per minimizzare tempo di turn-around medio, però se i job arrivano in tempi diversi potrebbe non essere ottimale. (Dipende dal quanto di tempo iniziale del processo).
- **Shortest Remaining time next**: Versione con prelazione dello shortest job first. Sceglie sempre il tempo con il minor quanto di tempo rimasto. Anche in questo caso dobbiamo conoscre il tempo.

Scheduling in sistemi interattivi

Gli scheduling nei sistemi interattivi si basano sul dare risposte giuste velocemente con principio d'equità tra i processi:

- **Round Robin**: Lista ciclica, viene eseguito un processo alla volta, per un quanto di tempo definito, pois si passa al processo successivo. Una cosa importante è scegliere il quanto di tempo giusto per evitare overhead.

> ⓘ **Tradeoff del quanto di tempo**
>
> Quanto lungo --> Riduce l'overhead, ma peggiora la reattività.
> Quanto corto --> Maggiore overhead e riduzione dell'efficienza della CPU.

- **Scheduling a priorità**: Creiamo delle classi di priorità che contengono i processi sui quali è applicato il round robin.

> ⓘ **Sulle classi di priorità**
>
> Dobbiamo tenere conto di alcune cose quando lavoriamo con le priorità:

1. La priorità di un processo può variare.
2. Bisogna evitare processi a priorità 0 (Starvation).
3. Due tipi di priorità: **Statica** (Non cambia nel tempo) e **Dinamica** (cambia nel tempo).
4. Permettere di usare la CPU anche a processi "bassi"

DEAMON

Processo in background che ci permette di ottnere delle risposte in maniera indiretta.

- **Shortest Job First**: Ottimizza il tempo medio di risposta nei sistemi batch.

Aging

Ogni volta che viene eseguito un processo viene fatto "invecchiare" in modo tale da poter ricalcolare il tempo rimasto di esecuzione. Ci serve per evitare la starvation dei processi. Praticamente **modifichiamo dinamicamente la priorità**.

- **Guaranteed scheduling:** Diamo la massima equità ad ogni processo.

Massima equità

Abbiamo n processi allora diamo $\frac{1}{n}$ della CPU ad ogni processo. per assicurasi il corretto funzionamento viene calcolato il rapporto tra il tempo di CPU consumato e dovuto per ogni processo.
Vengono eseguiti i processi con rapporto più basso finché non supera il concorrente più vicino.

- **Scheduling a lotteria:** Vengono assegnati dei biglietti della lotteria ai processi, poi viene effettuata un'estrazione per decidere quale processo prende la risorsa. Possiamo anche creare una priorità dando più biglietti a processi più importanti.

VANTAGGI

- Reattività.
- Cooperazione tra processi.
- Usato in situazioni in cui altri algoritmi non funzionano.

- **Scheduling fair-share**: Ad ogni utente viene assegnata una parte del processore sulla quale eseguire i sui processi, lo scheduler si assicura che ogni utente riceve la sua parte di processore indipendentemente dal numero di processi che possiede.

SCHEDULING IN SISTEMI REAL-TIME

Sono sistemi operativi critici che devono rispettare vincoli temporali, qualsiasi tipo di ritardo è un problema.

Possono essere di due tipi:

- **Hard**: Le scadenze devono essere rispettate.
- **Soft**: Qualche scadenza è tollerabile.

Sono usati per la costruzione di: Droni, missili, aerei, macchinari medici, ...
I processi devono essere brevi e prevedibili, nonché noti in anticipo.

Abbiamo due tipi di eventi:

- **Periodici**
- **Non Periodici**

La CPU deve riuscire a gestire la somma totale del tempo richiesto dei processi. Il carico dei processi deve essere minore di 1 ed è sempre tenuto sotto controllo dal SO.

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

Algoritmi di scheduling possono essere:

- **Statici** --> Richiede la **perfetta** conoscenze delle esigenze e scandeze
- **Dinamici**

Processi e scheduling

Uno dei problemi principali degli scheduler è che non accettano input dai processi utente, questo può portare a decisioni sub-ottimali.

Per risolvere questa situazione dobbiamo separare il meccanismo di scheduling e la politica di scheduling.

> ⓘ **Esempio**

Immagina un sistema con un meccanismo di scheduling che utilizza un algoritmo **Round-Robin** per alternare i processi, ma la politica di scheduling prevede che alcuni processi (ad esempio, quelli di sistema o di alta priorità) possano essere eseguiti prima di altri. In questo caso:

- **Il meccanismo** (Round-Robin) si occupa di determinare quando ogni processo deve essere eseguito, ma la **politica** definisce quali processi abbiano la priorità.

Abbiamo visto inoltre quesi algoritmi applicati unicamante ai processi, ma cosa succede nei thread?

Abbiamo due situazioni:

- **Thread a livello utente** --> Viene eseguito ogni thread di un processo prima di passare al prossimo, quindi abbiamo poche istruzioni eseguibili solo su un processo.
- **Thread a livello kernel** --> Il kernel seleziona un thread specifico per l'esecuzione, quindi abbiamo una flessibilità maggiore (possiamo spostarci su thread di diversi processi), ma a causa del cambio di contesto è piu lento.