Scheduler

Lo scheduler si occupa di **assegnare processi e thread alla CPU**, è una scelta molto importante per il sistema, poiché vogliamo massimare l'uso della CPU e minimizzarne il tempo d'accesso.
Esistono diversi algoriitmi di scheduling in base al sistema operativo che usiamo.

> **Storia dello scheduling**
>
> I primi scheduling dei sistemi batch erano lineari, ma con l'avvento della multiprogrammazione (Più processi o utenti da eseguire contemporanemante) è nata la necessità di avere degli schemi di scheduling più **complessi**.
> Diventato ancora più importanti con i personal computer dove l'interazione è costante. Non nasce solo per ottimizzare l'accesso al processo e per **ottimizare il tempo di esecuzione.**
> Quando usiamo un algoritmo di schedukling dobbiamo tenere conto di molti fattori anche in base all'hardware che stiamo usando.

Uno dei problemi principali con lo scheduler riguarda il **context switch**, molto costoso a fattore di tempo, bisogna trovare un tradeoff tra tempo di utilizzo del processore e tempo usato per lo scambio dei processi.

Possiamo distinguere i processi in:

- **Compute-bound** --> Stressano per lungo periodo il processore.
- **I/O-bound** --> Brevi tempi di attesa dell'I/O (Aspetto l'input dall'esterno). Sono tali a causa della bassa necessità di calcoli e non della durata delle richieste di CPU.

> **Esempio dell'importanza dell'hardware con scheduler**
>
> Immaginiamo un laptop, una caratteristica principale è che deve mantenere la batteria per lunghi periodi attiva.
> Quindi il processore nei momenti "morti" (EX: Tra un interrupt e un altro) può diciamo riposare risparamiando sulla batteria.

I processi tendono ad essere I/O bound e lo scheduler si trova fra lo stato di ready e attivo

Quando viene usato lo scheduler

Le situazioni in cui viene usato lo scheduler sono le seguenti:

- **Creazione del processo**: Decide se eseguire il processo padre o figlio.
- **Uscita di un processo**: Se un processo esce deve scegliere quello successivo.
- **Blocco del processo**: Se un processo si blocca bisogna sceglierne uno nuovo.
- **Interrupt di I/O**: Alla conclusione di un I/O un processo potrebbe diventare pronto.

> ⚠ **Quando faccio una fork il processo eseguito viene scelto sempre dallo scheduler. L'utente non sceglie nulla.**

Prelazione e tipi di scheduling

Prelazione: Interruzione forzata (Il processore caccia il processo) ed è imposta dall'esterno.

I processi possono essere anche:

- **NON preemtive (senza prelazione)**: Processo continua la sua esecuzione fino al completamento o fino a quando non è interrotto per motivi specifici (blocco, richiesta di I/O).
- **preemptive**: Il sistema operativo interrompe l'esecuzione di un processo per permettere ad altri processi di avere la CPU, garantendo che **nessun processo monopolizzi la risorsa CPU** per troppo tempo.

La prelazione è estremamente importante per garantire:

- **Migliore reatttività del SO**: Nessun processo blocca per troppo tempo la CPU.
- **Gestire le risorse in maniera efficiente**.
- **Garantire equità tra i processi**.

> ⓘ **Il clock definisce i quanti di tempo in cui possono essere eseguiti i nostri processi.**

Scenari di scheduling

Esistono tre ambienti principali di scheduling:

- **BATCH**: Algoritmi per attività periodiche, senza prelazione e priorità a prestazioni efficienti.
- **INTERATTIVO**: La prelazione è fondamentale, inoltre tutti i processi devono evolvere nel clock e nessun processo deve essere lasciato indietro.
- **REAL TIME**: Prelazione non sempre necessaria.

Obiettivi generali per gli algoritmi di scheduling

Sistemi Batch ---> Throughput, Tempo di Turnaround, Uso costante della CPU.

> ⓘ **Significati**
>
> Throughput ---> Aumentare il numero di job per un tempo fissato.
> Tempo di Turnaround ---> Minimizzare il tempo dallo start all'end di uno job.
> Uso costante della CPU.

Sistemi interattivi ---> Rispondere rapidamente agli utenti e soddifarne le aspettative in termini di tempi di risposta.

Real time ---> Rispetto delle scadenze e assicurarci che il funzionamento sia costante.

Inoltre ricordiamo che in tutti i sistemi è necessario avere:

- **equità**: garantire un'equa condivisione della CPU.
- **Impostazione della policy**.
- **Bilanciamento**: Mantenere tutti i componenti attivi.

La prima importante per evitare processi "morti", cioè che stanno solo in attesa.

> ⓘ **Altre informazioni:**
>
> Nei sistemi batch è ideale combinare processi CPU-bound e I/O-bound.
> Nei sistemi real-time è cruciale rispettare le scadenze.