Concetti basilari di scheduling

Index

Introduction

Un sistema operativo deve allocare risorse tra diversi processi che ne fanno richiesta contemporaneamente. Tra le diverse possibili risorse, c'è il tempo di esecuzione, che viene fornito da un processore. Questa risorsa viene allocata tramite lo **scheduling**

Scopo dello scheduling

Dunque lo scopo dello scheduling è quello di assegnare ad ogni processore i processi da eseguire, man mano che i processi stessi vengono creati e distrutti. Tale obiettivo va raggiunto ottimizzando vari aspetti:

- tempo di risposta
- throughput
- efficienza del processore

Obiettivi dello scheduling

L'obiettivo è dunque quelli di distribuire il tempo di esecuzione in modo equo tra i vari processi, ma al tempo stesso anche gestire le priorità dei processi quando necessario (es. vincoli di real time ovvero eseguire operazioni entro un certo tempo).
Deve inoltre evitare la starvation dei processi, ma anche avere un **overhead** basso, ovvero avere un tempo di esecuzione dello scheduler stesso basso

---

Tipi di Scheduling

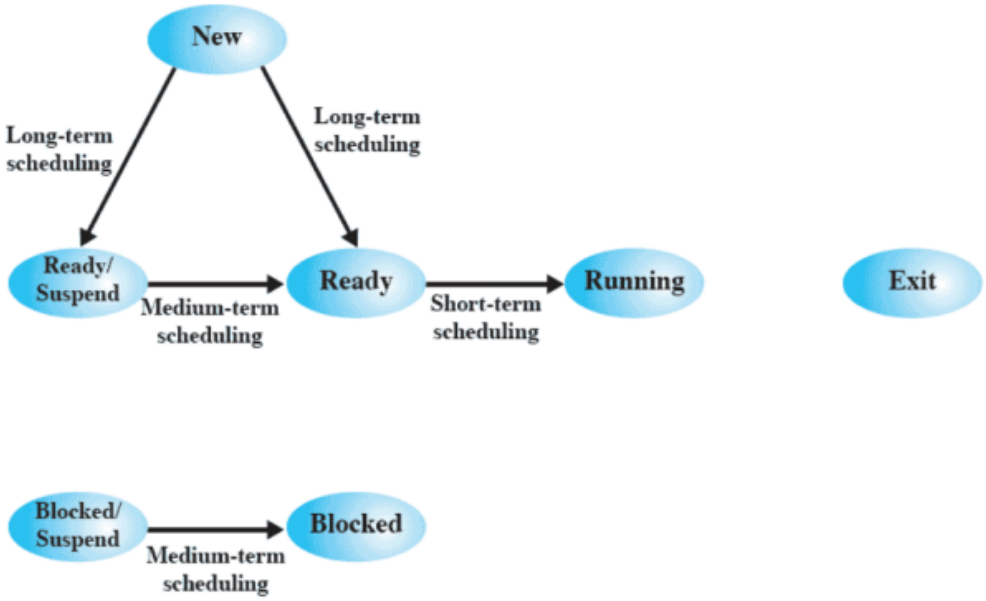
Esistono vari tipi di scheduler in base a quanto spesso un processo viene eseguito:

- *Long-term scheduling* → decide l'aggiunta ai processi da essere eseguiti (eseguito molto di rado)
- *Medium-term scheduling* → decide l'aggiunta ai processi che sono in memoria principale (eseguito sempre non molto spesso)
- *Short-term scheduling* → decide quale processo, tra quelli pronti, va eseguito da un processore (eseguito molto spesso)
- *I/O scheduling* → decide a quale processo, tra quelli con una richiesta pendente per l'I/O, va assegnato il corrispondente dispositivo di I/O

Processi e scheduling

Stati dei processi

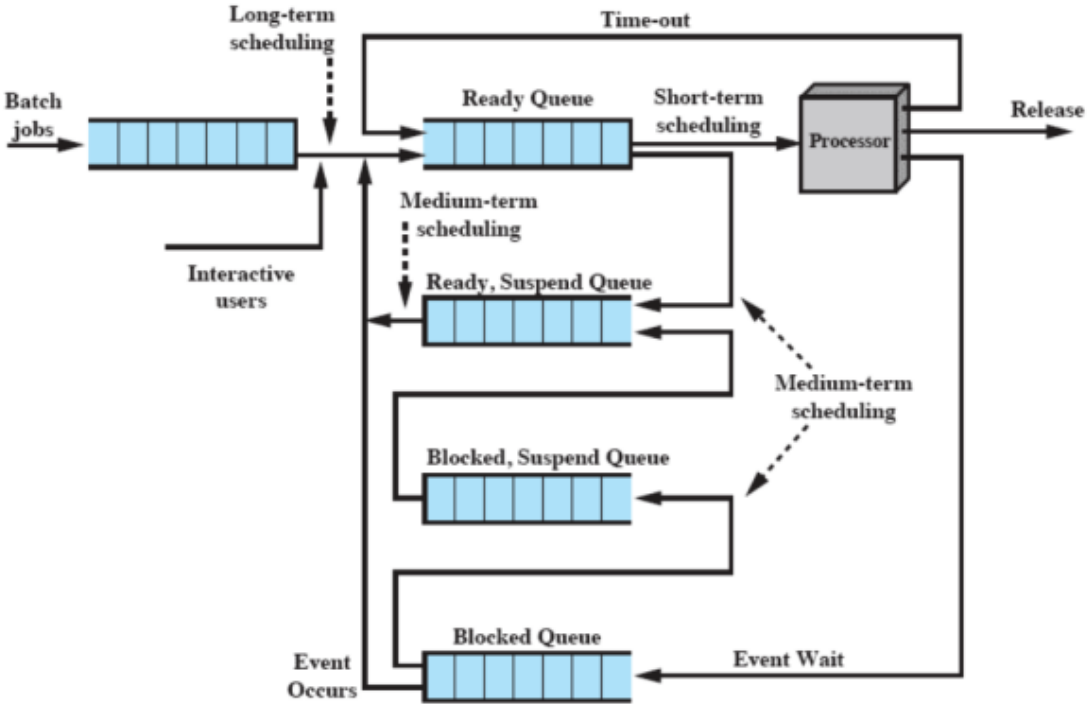
Ricordando il modello a 7 stati, le transizioni tra i vari stati sono decise dallo scheduler



Il long-term scheduler si occupa della creazione dei nuovi processi
Il mid-term scheduler si occupa dello swapping tra memoria principale e disco (e viceversa)
Lo short-term scheduler si occupa di decidere quali processi ready devono essere running

Modello a 7 stati >

Code dei processi



Long-term scheduling

Il long-term scheduling decide quali programmi sono ammessi nel sistema per essere eseguiti. Questo tipicamente è FIFO (first in first out) dunque il primo che è arriva è il primo ad essere ammesso, seppur tenga conto di criteri come priorità, requisiti per I/O ecc.
Controlla quindi il grado di multiprogrammazione, e all'aumentare del numero di processi, diminuisce la percentuale di tempo per cui ogni processo viene eseguito

Tipiche strategie

- i lavori batch (non interattivi) vengono accodati e il LTS li prende man mano che lo ritiene "giusto"
- i lavori interattivi vengono ammessi fino a "saturazione del sistema"
- se si sa quali processi sono I/O-bound e quali CPU-bound (quali usano più la cpu e quali più l'i/o) mantiene un giusto mix tra i due tipi; oppure se si sa quali processi fanno richieste a quali dispositivi di I/O, fare in modo da bilanciare tali richieste

In generale si può dire che il LTS viene chiamato in causa quando vengono creati dei processi ma ad esempio interviene anche quando termina un processo o quando alcuni processi sono idle da troppo tempo

---

Medium-term scheduler

Il medium-term scheduler è parte integrante della funzione di swapping per i processi. Il passaggio da memoria secondaria a principale è basato sulla necessità di gestire il grado di multiprogrammazione

---

Short-term scheduler

Lo short-term scheduler è chiamato anche *dispatcher*, è quello eseguito più frequentemente, ed è invocato sulla base di eventi che accadono:

- interruzioni di clock
- interruzioni di I/O
- chiamate di sistema
- segnali

Scopo

Lo scopo dello short-term scheduler è quello di allocare il tempo di esecuzione su un processore per ottimizzare il comportamento dell'intero sistema, dipendentemente da terminati indici prestazionali. Per valutare una data politica di scheduling, occorre prima definire dei criteri

Criteri

Occorre distinguere tra criteri per l'utente e criteri per il sistema prestazionali e non prestazionali.
Quelli prestazionali sono quantitativi e facili da misurare
Quelli non prestazionali sono qualitativi e difficili da misurare

Criteri utente

Prestazionali:

- **Turnaround time**
- **Response time**
- **Deadline**

Non prestazionali:

- **Predictability**

Turn-around Time

Tempo per la creazione di un processo (sottomissione) e il suo completamento, compresi tutti i vari tempi di attesa (I/O, processore). Spesso viene utilizzato per processi batch (non interattivi)

Response time

Qui si trattano quasi esclusivamente processi interattivi. Si tratta del tempo che intercorre tra la sottomissione di una richiesta (l'utente fa un'interazione) e l'inizio del tempo di risposta.
Per il response time lo scheduler ha un duplice obiettivo

- minimizzare il tempo di risposta medio
- massimizzare il numero di utenti con un buon tempo di risposta

Deadline e Predictability

Per deadline si intende, nei casi in cui un processo specifica una scadenza, lo scheduler dovrebbe come prima cosa massimizzare il numero di scadenze rispettate
Per predictability si intende che non deve esserci troppa variabilità nei tempi di risposta e/o di ritorno, a meno che il sistema non sia completamente saturo

Criteri sistema

Prestazionali:

- **Throughput**
- **Processor utilization**

Non prestazionali

- **Fairness**
- **Enforcing priorities**
- **Balancing resources**

Throughput

Ricordiamo il throughput è il numero di processi in grado di eseguire in una determinata unità di tempo, e il dispatcher ha il compito di massimizzarlo (nonostante in parte dipenda anche dal tempo richiesto dal processo)

Processor utilization

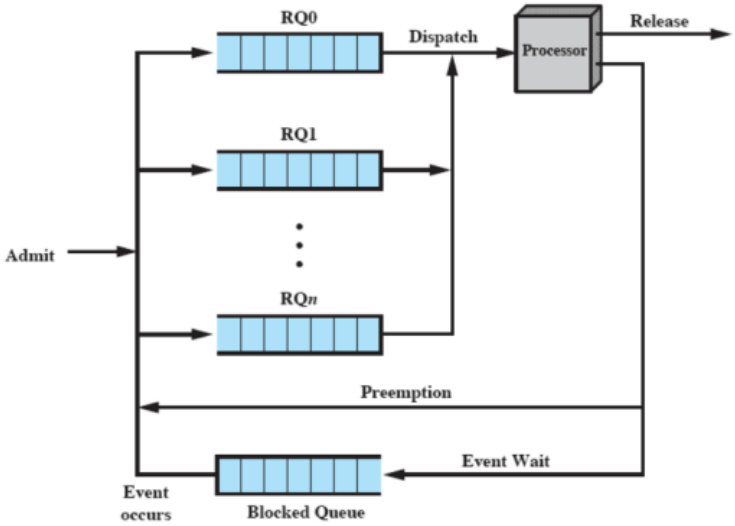
Il processor utilization è la percentuale di tempo in cui il processore viene utilizzato, si cerca dunque di minimizzare il tempo in cui il processore è idle
Particolarmente utile per sistemi costosi, condivisi tra più utenti

Bilanciamento delle risorse

Lo scheduler deve far si che le risorse del sistema siano usate il più possibile. Infatti i processi che useranno meno le risorse attualmente più usate dovranno essere favoriti

Fairness e priorità

Se non ci sono indicazioni dagli utenti o dal sistema (es. non c'è priorità), tutti i processi devono essere trattati allo stesso modo, dunque a tutti i processi deve essere data la stessa possibilità di andare in esecuzione (evitando la starvation).
Se invece ci sono priorità, lo scheduler deve favorire i processi a priorità più alta (occorre avere più code, una per ogni livello di priorità)



Potrebbe però nascere un problema: un processo con priorità più bassa potrebbe soffrire di starvation; la soluzione sta nell'aumentare la priorità del processo a mano a mano che l'"età" del processo aumenta