Schedulazione

Schedulazione

Obiettivo

L'obiettivo centrale della schedulazione dei processi e' la gestione della turnazione dei processi sul processore. Questo implica la definizione di politiche di ordinamento dei processi per gestire tale turnazione.

Livelli di schedulazione

La schedulazione puo' essere attivata a vari livelli:

- A breve termine (<1s);
- A medio termine;
- A lungo termine (>1m).

Schedulazione a breve termine

Detta anche **Short-term scheduler** o *CPU scheduler*, ordina i processi che sono:

- Gia' presenti in memoria centrale;
- Nello stato di *Ready-To-Run*.

Il processo posto in prima posizione dall'ordinamento e' quello che il dispatcher mettera' in esecuzione quando avverra' il cambiamento di contesto successivo.

La schedulazione a breve termine deve essere eseguita abbastanza frequentemente in modo tale che i processi vedano evolvere la propria computazione in modo parallelo. E' tipico avere questo tipo di schedulazione ogni 100ms.

Gli algoritmi utilizzati sono usualmente semplici per poter essere eseguiti rapidamente senza sovraccaricare la gestione.

Schedulazione a medio termine

Detta anche **Medium-term scheduler**, ha come obiettivo quello di mediare le caratteristiche tra gli schedulatori a breve e lungo termine. Sorgono pero' alcuni problemi:

- Le prestazioni del sistema non sono ottimali a causa dell'alta concorrenza per l'uso del processore;
- Lo sfruttamento del processore non e' ottimale a causa di una distribuzione sbilanciata tra processi CPU-bond e I/O-bond;
- La memoria ventrale viene esaurita a causa delle dimensioni dei processi caricati e delle creazione dinamica di variabili;
- La memoria centrale disponibile scarseggia per ciascun processo con un conseguente sovraccarico di gestione della stessa.

Per poter:

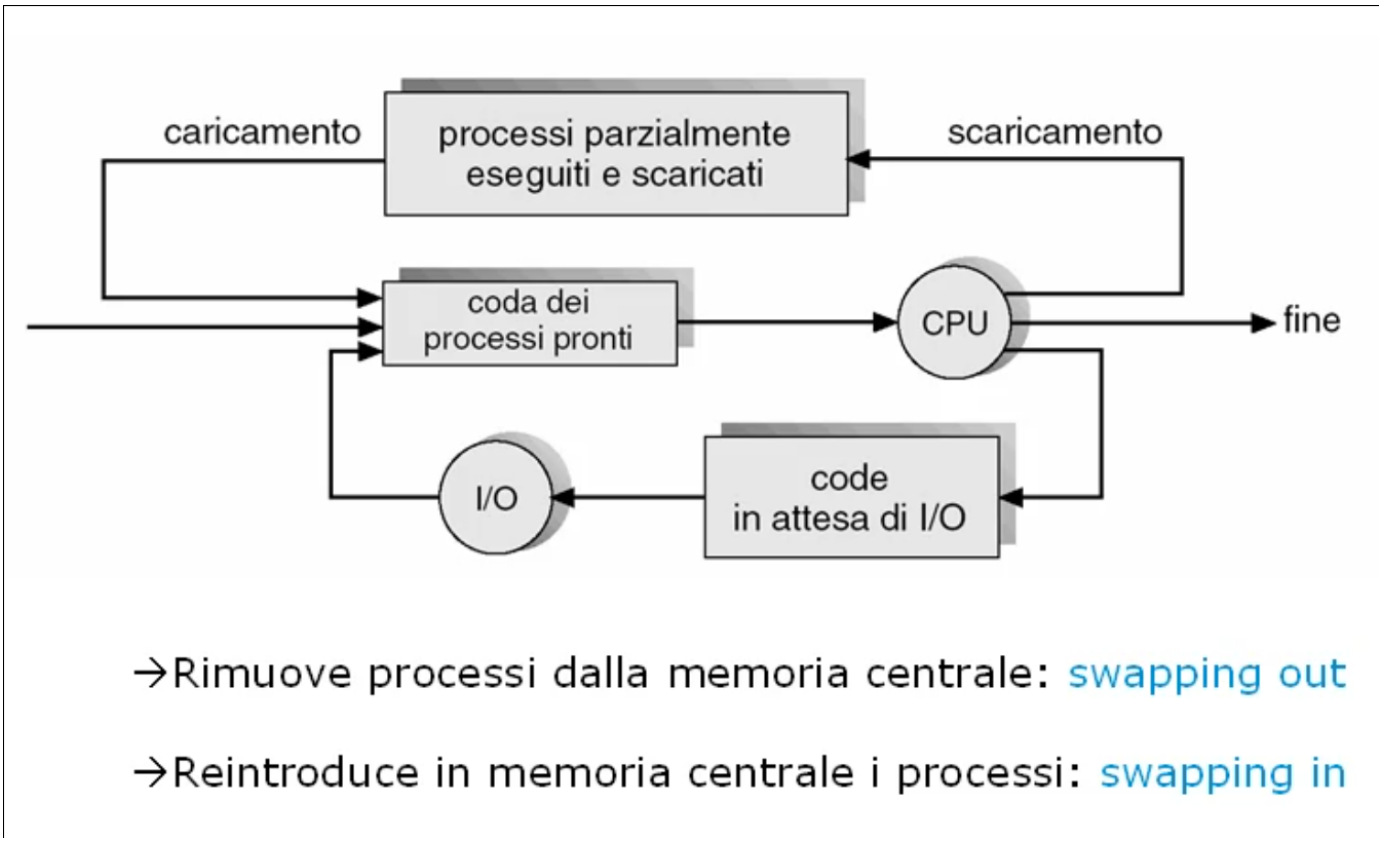
- Ridurre la concorrenza tra i processi;
- Ottimizzare la distribuzione dei processi CPU-bound e I/O-bound;

- Dare maggiore memoria centrale ai processi caricati in essa.

E' possibile modificare dinamicamente il gruppo di processi caricati in memoria centrale e posti nello stato di *Ready-To-Run* dallo schedulatore a lungo termine. Questo consente di adattare il gruppo di processi caricato in memoria centrale all'effettivo carico di lavoro.

Effettivmante quindi lo schedulatore a medio termine ordina i processi che erano stati selezionali dallo schedulatore a lungo termine per il caricamento in memoria centrale e l'ammissione allo stato di pronto all'esecuzione:

- Ponendo realmente in memoria centrale solo alcuni dei processi *Ready-To-Run* in modo da garantire un bilanciamento ottimale effettivo tra CPU-bound e I/O-bound e un sovraccarico minimo per la gestione della memoria centrale;
- Lasciando in un'area di memoria di massa temporanea gli altri processi del gruppo selezionato dallo schedulatore a lungo termine.



Schedulazione a lungo termine

Detta anche **Long-term scheduler** o *Job scheduler*, ordina i processi attivati nel sistema identificando il gruppo di processi che devono essere:

- Caricati in memoria centrale per l'esecuzione;
- Posti nello stato di pronto all'esecuzione.

Il Job scheduler costruisce il gruppo mescolando processi CPU-bound e I/O bound in modo da massimizzare lo sfruttamento atteso del processore.

Uno schedulatore a lungo termine puo' essere anche del tutto assente in un sistema o essere davvero minimale. Fa uso di algoritmi complessi perche' deve tener conto della predizione del comportamento dei processi e quindi eseguire un bilanciamento ottimale tra le varie esigenze.

Deve essere eseguito poco frequentemente per non sovraccaricare il sistema (tipicamente una volta ogni qualche minuto).

Attivazione

Non pre-emptive scheduling

Quando la schedulazione viene attivata in pre-rilascio, tramite rilascio forzato.

Il processo in esecuzione viene cambiato:

- Dopo aver richiesto un'operazione di I/O;
- Dopo aver creato un processo e ne attende la terminazione;
- Quando rilascia volontariamente il processore;
- Quando termina.

La schedulazione viene attivata in maniera sincrona rispetto all'evoluzione della computazione del processo.

Pre-emptive scheduling

La schedulazione viene attivata in maniera asincrona perche' il processo in esecuzione puo' essere cambiato anche quando scade il *time slice* e questo e' tipico dei sistemi *time-sharing*.

Criteri di valutazione per la schedulazione

Criteri di schedulazione

Per valutare la bonta' di una politica di turnazione possiamo adottare diversi criteri che mettono in evidenza diversi aspetti di uso del processore e dell'evoluzione della computazione dei processi:

- Utilizzo della CPU: percentuale di tempo per cui la CPU esegue computazione utile;
- Capacita' di frequenza di completamento dei processi (**throughput**): numero di processi completati nell'unita' di tempo;
- Tempo di completamento dei processi (**turnaround time**): per i processi in cui la terminazione di un processo deve avvenire entro un tempo massimo;
- Tempo di attesa della CPU;
- Tempo di risposta dei processi.

Obiettivi di ottimizzazione

Avere un buon algoritmo per la schedulazione vuol dire cercare di ottimizzare i criteri di valutazione.

E' inoltre importante cercare di minimizzare la varianza dei parametri caratteristici, per garantire la predicibilita' del comportamento del sistema.

Metodi di valutazione

Valutazione analitica

Si fa uso di una modellazione deterministica in cui vengono costruite le formule dell'algoritmo di schedulazione.

Viene calcolato il valore del parametro caratteristico semplicemente sostituendo la distribuzione dei carichi di lavoro come ingressi della formula ottenuta. Avremo cosi' un risultato semplice, veloce e preciso.

E' necessario pero' conoscere a priori una quantificazione esatta dei carichi di lavoro e questo non e' sempre possibili. Inoltre i risultati possono essere difficilmente generalizzati e cio' limita l'astrazione.

Valutazione statistica

Modelli a reti di code

Il sistema viene descritto come un insieme di servizi e ogni servizio ha una sua coda di attesa per i processi.

Le transizioni tra le code rappresentano i flussi delle richieste di servizio.

Analizzare le reti di code vuol dire:

- Specificare la topologia del grafo della rete;
- Specificare le caratteristiche di ogni servizio:
  - Frequenza d'arrivo delle richieste;
  - Tempo di servizio.
- Analisi della rete basata sulla **Teoria delle code**,

Il risultato dell'analisi statistica valuta:

- L'utilizzo della risorsa;
- La lunghezza media della coda;
- Il tempo medio d'attesa.

Sono necessarie alcune semplificazioni per poter riuscire a realizzare la coda e risolvere con le tecniche matematiche il modello cosi' costruito.

Simulazione

Possiamo realizzare un modello software del sistema reale che vogliamo ottenere (includendo lo schedulatore, i processi applicativi, ecc) in modo tale da simularne l'esecuzione.

Questo vuol dire identificare alcuni dati significativi che caratterizzano le attivita' dei processi e applicarle all'interno della simulazione. In questo ambiente sintetico possiamo misurare le caratteristiche della schedulazione assumendo un comportamento statistico del nostro sistema (variando la distribuzione dei dati su cui i vari processi operano).

Implementazione

Se vogliamo avere una valutazione accurata, spesso un'analisi di tipo statistico non basta, diventa dunque indispensabile andare a realizzare il sistema.

Bisogna dunque implementare il sistema HW/SW, sistema operativo e algoritmo di schedulazione con gli ingressi reali. Si misurando dunque le caratteristiche desiderate del nostro approccio di schedulazione.

Questo e' l'approccio piu' oneroso e richiede la cooperazione degli utenti che sono coinvolti in questa misurazione.

La schedulazione viene scelta in base alle caratteristiche reali e questi carichi di lavoro possono essere raccolti ed analizzati da parte di componenti del sistema operativo. Questo e' importante perche' ci permette di avere un meccanismo di raffinamento della schedulazione durante la vita operativa del sistema.

Politiche di schedulazione

Politiche e algoritmi di schedulazione

First Come, First Served

Abbiamo una coda in cui vengono inseriti i processi *Ready-To-Run* e vengono estratti secondo l'ordine di arrivo nella coda.

E' un algoritmo **non pre-emptive**, non porta alla sospensione del processo in esecuzione per l'attivazione dell'algoritmo di schedulazione.

Solo quando il processo in esecuzione rilascia il processore allora il primo processo in attesa andra' ad usare il processore.

Il problema di questo algoritmo sono:

- I processi CPU-bound possono monopolizzare il processore;
- Il tempo di attesa puo' essere alto in base all'ordine di arrivo dei processi.

Shortest Job First

E' un algoritmo che cerca di ridurre il tempo di attesa rispetto all'algoritmo *First Come, First Served*.

I processi entrano nella coda secondo il loro ordine di arrivo e qui vengono ordinati in base al loro *costo* in unita' di tempo.

Puo' essere realizzato in due modi:

- **Pre-emptive**: quando un processo diventa *Ready-To-Run* provoca la sospensione del processo in esecuzione e l'attivazione dell'algoritmo di schedulazione, in modo tale da vedere quale dei processi ha piu' diritto di usare il processore;
- **Non pre-emptive**: quando un processo diventa *Ready-To-Run* non comporta l'interruzione del processo in esecuzione per richiedere l'esecuzione dell'algoritmo di schedulazione. Solo quando il processo in esecuzione ha terminato la sua evoluzione della computazione, verra' eseguito l'algoritmo di schedulazione.

Nel caso di realizzazione in modalita' *pre-emptive* il tempo medio di attesa puo' decrescere.

L'algoritmo garantisce il tempo minimo di attesa nel momento in cui e' noto a priori il tempo richiesto da ciascuno dei processi. Ovviamente questo parametro e' difficile da conoscere e quindi e' possibile usare delle tecniche di predizione (ad esempio: una stima statistica basata sul tipo di uso che ha fatto il processo in precedenza, o tramite una media esponenziale).

Priorita'

Questo approccio valorizza l'importanza relativa dei processi piuttosto che il loro ordine di arrivo. L'importanza relativa di un processo viene indicata mediante un **indice di priorita'**. Si possono intendere gli indici di priorita' secondo:

- Logica diretta: indice di priorita' alto -> Priorita' alta;
- Logica inversa: indice di priorita' basso -> Priorita' alta.

Questo algoritmo puo' essere realizzato in modo:

- **Pre-emptive**: quando un processo diventa *Ready-To-Run* interrompe il processo in esecuzione chiedendo la schedulazione;
- **Non pre-emptive**: quando un processo diventa *Ready-To-Run* non interrompe il processo in esecuzione per richiedere la schedulazione ma attende il suo completamento.

Il problema che si verifica in questo tipo di gestione e' che i problemi con priorita' piu' bassa potrebbero venire bloccati indefinitamente (**starvation**).

Per ovviare al problema si puo' introdurre un progressivo invecchiamento della prioria' (**aging**) con periodico ripristino al valore iniziale oppure ringiovanimento fino al valore iniziale.

Round Robin

Si tratta di un algoritmo che effettua una vera e propria rotazione dei processi in esecuzione rispetto all'ordine di arrivo e allo stato del processo.

E' un approccio tipico nei sistemi time-sharing.

E' simile alla politica *First Come, First Served* con aggiunta di *pre-emption*.

L'algoritmo da' una distribuzione uniforme del tempo di elaborazione tra i vari processi pronti.

La velocita' di esecuzione dei processi **dipende fortemente dal numero dei processi pronti** nella coda.

Il tempo turnaround dipende dalla durata del *time-slice*:

- *Time-slice* molto lungo: l'algoritmo *Round Robin* tende a diventare *First Come, First Served*;
- *Time-slice* molto breve: l'algoritmo *Round Robin* portera' ad una forte condivisione del processore, dove ogni processo vede 1/N capacita' computazionale ad esso assegnata. Il problema e' che questo porta a frequenti cambiamenti di contesto e quindi a sovraccarichi di gestione;
- L'ideale empirico consisterebbe nel completare circa l'80% dell'elaborazione richiesta dal processo in un unico *time-slice*.

Coda a piu' livelli

I processi sono raggruppati per tipologie omogenee e ogni tipologia di processo e' assegnata ad uno specifico livello di schedulazione rappresentato da una coda di attesa specifica per l'uso del processore.

Per ogni coda e' possibile introdurre un algoritmo specifico di schedulazione.

L'insieme delle code verra' poi schedulato da un algoritmo dedicato: usualmente vengono usate delle schedulazioni **pre-emptive** a priorita' fisse.

Coda a piu' livelli con reatroazione

E' una variante della coda a piu' livelli. In questo caso i processi possono migrare da un livello all'altro (dunque da una coda ad un'altra) attivando un meccanismo di promozione o degradazione.

La coda a priorita' piu' elevata puo', ad esempio, avere un *time-slice* piu' breve per garantire un'alta rotazione dei processi. Man mano allunghiamo il *time-slice* per garantire una rotazione piu' lenta e quindi doverci occupare di un inferiore tempo di gestione per processi meno rilevanti.

Si hanno quindi code di attesa separate in funzione dell'uso dinamico che si vuol far fare ai processi.

Si ha un algoritmo specifico di schedulazione per ogni coda e in piu' si hanno delle politiche di allocazione dei processi quando nascono nella coda opportuna e di promozione/degradazione per spostarli a livelli superiori/inferiori.

Schedulazione in sistemi multiprocessore

Nel caso di sistemi multiprocessore, la schedulazione deve considerare le caratteristiche dell'architettura del sistema di elaborazione:

- Processori omogenei/eterogenei;
- Memoria solo condivisa / anche locale;
- Periferiche ugualmente accessibili da tutti i processori / da un singolo processore.

Si possono pensare delle politiche di schedulazione specifiche a seconda della combinazione di queste caratteristiche HW, ad esempio:

- Con processori omogenei, memoria solo condivisa e periferiche accessibili da tutti: si puo' avere una coda unica per tutti i processi;
- Con processori omogenei, memoria anche locale e periferiche accessibili da tutti: la coda specifica per ogni processore puo' stare sia nella coda specifica che nella memoria locale specifica dei processori;
- Con processori omogenei, memoria solo condivisa e periferiche accessibili da alcuni: una coda per ogni processore che gestisce la specifica periferica d'interesse;
- Con processori omogenei, memoria anche locale e periferiche accessibili da alcuni: una coda per la gestione dei processori puo' essere messa nella sua memoria locale;
- Con processori eterogenei: una coda per ogni processore o una coda per un gruppo di processori omogenei.

Si possono pensare, nel caso di sistemi multiprocessore, diversi tipi di **multiprocessamento**:

- **Asimmetrico**: un processore master che esegue il sistema operativo che provvede a gestire la schedulazione di tutti i processi e processori slave che eseguono solo i processi applicativi;
- **Simmetrico**: ogni processore esegue il sistema operativo e provvede a schedulare i vari processi applicativi.

Politiche di schedulazione per sistemi in tempo reale

Sistemi in tempo reale

Sistemi in tempo reale stretto

I sistemi in tempo reale stretto (**Hard Real-Time System**) sono quelli in cui e' obbligatorio che un processo termini la sua computazione entro un tempo massimo garantito dalla sua attivazione.

Per gestire la schedulazione in questo tipo di sistemi esistono alcuni approcci specifici a seconda della tipologia del sistema.

Schedulazione con completamento di tempo garantito

L'algoritmo di schedulazione con tempo massimo garantito prevede che lo schedulatore possa:

- Accettare un processo garantendone il completamento entro il tempo massimo consentito;
- Rifiutare il processo.

L'accettazione del processo e' basata sulla possibilita':

- Di stimare il tempo di completamento del processo;
- Di prenotazione delle risorse necessarie al processo.

Le politiche di schedulazione che si possono adottare sono quelle tradizionali.

L'aspetto essenziale e' che, in base al carico di lavoro correntemente accettato e quello previsto per il nuovo processo, si effettui un'analisi a priori che porta all'accettazione o al rifiuto del nuovo processo. Il problema e' la **predicibilita'** del tempo di completamento.

Schedulazione di processi periodici

Molti processi, ad esempio nel campo dell'automazione industriale, sono di tipo periodico.

Un processo P_i avra' dunque le seguenti caratteristiche:

- Un tempo fisso di elaborazione t_i;
- Una scadenza d_i;
- Una periodicita' p_i.

Tipicamente questi tempi vengono ordinati in modo crescente:

> $0 <= t_i <= d_i <= p_i$

Nel caso di processi periodici possiamo utilizzare, come politiche di schedulazione:

- Round Robin;
- Priorita' assegnata in base alla scadenza d_i o alla frequenza $1/pi$ di attivazione del processo.

Il periodo e' specifico rispetto al processo considerato. Altri processi possono avere periodi differenti.

In fase di ammissione viene effettuato un controllo circa la possibilita' di completamento entro la scadenza dichiarata con la politicha di schedulazione adottata.

Schedulazione a frequenza monotona

Un altro algoritmo utilizzato per processi periodici basati su priorita' con meccanismi di pre-emption e' la schedulazione a frequenza monotona.

In questo sistema il tempo di elaborazione e' omogeneo per ogni iterazione del processo P_i.

La priorita' puo' essere assegnata:

- Usando delle tecniche statiche;
- In maniera proporzionale alla frequenza $1/p_i$.

Questa tecnica di schedulazione prevede la pre-emption: se un processo a piu' alta priorita' di esecuzione diventa *Ready-To-Run* e quello che sta venendo eseguito in quell'istante ha una priorita' inferiore, quest'ultimo viene sospeso per rilasciare il processore.

Schedulazione a scadenza piu' urgente

La schedulazione **Earliest-Deadline First** (EDF) puo' essere applicata sia per processi periodici che per processi non-periodici. Inoltre il tempo di elaborazione dei processi non deve piu' necessariamente essere omogeneo.

Questa tecnica prevede che si assegnino delle prioria' ai processi in maniera:

- Inversamente proporzionale alla loro scadenza d_i;
- Dinamica, in funzione dei processi che diventano *Ready-To-Run*.

Sistemi in tempo reale lasco

Nei sistemi **Soft Real-Time System** i processi non sono tutti critici e prioritari, ma solo alcuni possono essere processi critici e prioritari per i quali e' necessario operare in un predeterminato periodo di tempo.

Possiamo dunque utilizzare una schedulazione a priorita' separando i processi critici da quelli che non lo sono. La priorita' e':

- Statica per i processi critici;
- Eventualmente dinamica per i processi non critici.

Il fattore importante in questi sistemi e' avere una **bassa latenza** di dispatch:

- Possibilita' di interrompere le chiamate di sistema lunghe, con dei punti specifici dove effettuare la pre-emption;
- Tramite kernel interrompibile.

Schedulazione dei thread

Livelli di schedulazione

La schedulazione puo' essere effettuata in due livelli diversi.

Se il sistema operativo supporta in modo nativo i thread di un processo allora potremo avere la gestione della schedulazione direttamente a livello del sistema operativo

Alternativamente, se il sistema operativo mette semplicemente a disposizione delle librerie per gestire i thread all'interno del processo applicativo, avremo la gestione della schedulazione che dovra' essere fatta a livello del singolo processo. Il sistema operativo vedra' schedulato l'intero processo e non i singoli thread.

Schedulazione nel processo

La schedulazione a livello di processo (**Thread user-level scheduling**) prevede una visione a livello di contesa delle risorse all'interno del processo (**Process-Contention Scope - PCS**). Questo e' gestito dalla libreria di schedulazione fornita all'interno della libreria dei thread (fornita dal sistema operativo).

In questo caso abbiamo un insieme di processi divisi in un insieme di thread che vengono eseguiti sul sistema operativo.

Lo scheduling dei thread e' quindi interno ai processi, mentre il sistema operativo effettuera' la schedulazione "globale" dei processi.

Le schedulazioni che vengono eseguite nelle librerie fornite dal sistema operativo per i thread sono:

- A priorita' fisse o modificabili dal programmatore, con pre-emption;
- First Come, First Served;
- Round Robin.

Schedulazione di sistema

Nel caso di schedulazione a livello di sistema (**Thread kernel-level scheduling**) oppure si hanno i thread a livello user ma mappati su thread a livello di kernel (con o senza LWP) si ha la possibilita' di vedere i processi divisi per thread, e sara' il sistema operativo a gestire direttamente la schedulazione di ciascuno dei thread.

La visione della contesa per processore avviene a livello di sistema (**Process-Contention Scope - SCS**).