UD 3 - SCHEDULAZIONE

Nei sistemi operativi multiprogrammati più processi possono essere in esecuzione e tutti concorrono nell'uso delle risorse. Poiché solo un processo alla volta può accedere alla cpu, si è reso necessario innalzare il livello di astrazione per analizzare le tipologie di processi e mediante l'adozione di opportune politiche di schedulazione dei processi, sfruttare al meglio l'attività del processore.

I sistemi operativi possono schedulare secondo tre modalità, detti livelli di schedulazione: a breve termine, a lungo termine e a medio termine (i nomi stessi sottolineano la loro dipendenza dalla frequenza con cui sono eseguiti).

I processi pronti sono salvati in memoria centrale e lo schedulatore a breve termine sceglierà il processo che deve andare in esecuzione. Il dispatcher effettuerà il cambio di contesto e assegnerà la cpu.

I criteri di scelta adottati dallo schedulatore sono definiti da diverse tipologie di algoritmi. Il calcolo degli algoritmi di schedulazione è definita sulla base dei picchi di utilizzo della cpu, che possono essere più o meno frequenti a seconda se il processo è:

- **cpu-bound**, cioè con forte utilizzo della cpu
- **I/O bound**, cioè con brevi utilizzi di cpu e frequenti operazioni di I/O

Modelli

I dati da utilizzare per la definizione e la scelta dell'algoritmo costituiscono un modello e tale modellazione può essere:

- **Di tipo deterministico**. Si basa su dati in ingresso predeterminati ma essi devono essere precisi e le caratteristiche dei processi devono essere stabili, altrimenti il modello non è reale e quindi le stime sono molto approssimative.
- **A rete di code**. Si basa sui calcoli della media esponenziale dei picchi di CPU e I/O dell'esecuzione dei processi. Questi valori permettono di stimare i tempi di utilizzo e di attesa dei processi. Non rispecchiano la realtà e sono molto approssimati.
- **Tramite simulazione**. Vengono raccolti e analizzati i dati prodotti da un modello del sistema del computer che simula l'ambiente reale. La sua realizzazione è molto costosa ed è tanto più efficace quanto più lo si utilizza.
- **Tramite implementazione**. L'algoritmo è eseguito nel suo ambiente di destinazione, in un contesto di utilizzo reale. Richiede costose modifiche al sistema operativo e può avere un impatto negativo sull'utenza.

Criteri di valutazione

I criteri da osservare nella definizione e nella scelta di un algoritmo devono tenere conto dei seguenti fattori:

- Tempo di utilizzo della cpu
- Throughput (numero di processi completati per unità di tempo)
- Turnaround time (tempo che intercorre tra la creazione e il completamento di un processo)
- Tempo di attesa nella coda dei processi pronti
- Tempo di risposta (tempo che intercorre tra l'inizio dell'esecuzione del processo e la sua prima risposta se esso genera diversi output).

L'algoritmo ideale massimizza i primi due fattori e minimizza i successivi tre.

Politiche di schedulazione

Gli algoritmi possono seguire due schemi:

- non preemptive, cioè un processo detiene la cpu fino al passaggio nello stato di attesa o al termine;
- preemptive, cioè un processo può essere sospeso per permetterne l'esecuzione di un altro.

First Come First Served

Molto semplice da implementare, consiste in una coda FIFO (First In First Out) di processi. Si ha una coda in cui i processi vengono inseriti e dalla quale vengono estratti quando il processore si libera. Questo algoritmo è non pre-emptive, infatti non porta alla sospensione dei processi in esecuzione per l'attivazione dell'algoritmo di schedulazione (non adatto a sistemi time sharing).

Il problema è che i processi legati alla CPU possono monopolizzare il processore, mentre l'altro problema è il tempo di attesa che può diventare alto a seconda del momento in cui i vari processi entrano nella coda.

Shortest Job First

L’obiettivo quello di eseguire prima i processi più brevi per abbassare il tempo di attesa. Nella coda dei processi pronti, i processi saranno ordinati in ordine crescente di unità di tempo di elaborazione necessaria. Questo algoritmo può essere realizzato sia in modo pre-emptive sia in modo non preemptive.

Nel primo caso il processo che diventa pronto interrompe il processo in esecuzione richiedendo la schedulazione, mentre nel secondo caso il processo che diventa pronto non interrompe il processo in esecuzione ma lo lascia completare e solo successivamente viene eseguita la schedulazione. Globalmente, usando il modello implementato in maniera pre-emptive, il tempo di attesa medio può scendere rispetto allo stesso modello implementato in maniera non preemptive.

A priorità

A seconda delle caratteristiche dei processi e della loro importanza vengono loro assegnate delle priorità che influiranno sulla loro posizione in coda. Può essere di tipo preemptive o non preemptive, a seconda se un processo con priorità più alta possa o meno sospendere un processo in esecuzione.

Benché questo algoritmo assicuri il completamento dei processi più importanti, può dar luogo a starvation dei processi minori. Quest'ultimo problema può essere risolto con l'introduzione dell'aging del processo. Con l'aumentare del tempo d'attesa, aumenta la priorità del processo.

Round Robin

I processi sono accodati secondo l'algoritmo FCFS con modalità preemptive e viene definito un quanto di tempo della cpu di cui può disporre un processo. Allo scadere del quanto, il processo in esecuzione è sospeso e la cpu viene assegnata al successivo processo che la utilizzerà a sua volta solo per il quanto di tempo.

Questo algoritmo è tipico dei processi time-sharing perché fa avanzare tutti i processi progressivamente dando l'impressione che siano eseguiti in contemporanea.

E' molto importante la definizione del quanto di tempo, poiché un quanto troppo lungo annullerebbe i benefici della preemption e uno troppo breve avrebbe un forte impatto sul sistema e richiederebbe risorse adeguate a causa dei frequenti cambi di contesto.

Con coda a più livelli / feedback

I processi sono suddivisi in code a seconda della loro tipologia e ad ogni coda è assegnata una priorità. La tipica suddivisione è tra processi in primo piano e processi batch: i primi hanno una priorità maggiore rispetto ai secondi.

Ogni coda può utilizzare una schedulazione diversa e ad ogni coda può essere assegnata una percentuale di tempo della cpu. Con la variante feedback un processo può essere spostato in un'altra coda per prevenire situazioni di starvation o di forte utilizzo di cpu.

E' un algoritmo molto complesso da implementare poiché i criteri da utilizzare nelle varie tipologie di code sono molteplici.