4 - Scheduling

Introduzione allo Scheduling

In un sistema multiprogrammato molti processi e thread competono per la CPU, quindi è necessario scegliere a quale processo o thread assegnare la CPU. La parte del sistema operativo che effettua lo scheduling è detto scheduler e l'algoritmo che utilizza si chiama algoritmo di scheduling. Molti problemi di scheduling per processi valgono anche per thread, nel caso in cui i thread siano gestiti dal kernel.

Nei sistemi batch storici, lo scheduling era lineare, ovvero i job venivano eseguiti in modo sequenziale. Con la multiprogrammazione, lo scheduling è diventato complesso a causa della concorrenza tra utenti.

Costi in termini di tempo dello scheduling

Lo scambio di processi (context switch) è oneroso:

- Cambio da modalità utente a modalità kernel.
- Salvataggio dello stato del processo.
- Esecuzione dell'algoritmo di scheduling.
- Aggiornamento della MMU con la nuova mappa della memoria.
- Potenziale invalidazione della memoria cache.

Tutte queste operazioni possono consumare tempo alla CPU.

Problema di Scheduling dei Processi

Obiettivi

- **Equità**: garantire un'equa condivisione della CPU a tutti i processi.
- **Imposizione della policy**: garantire l'attuazione delle policy dichiarate.
- **Bilanciamento**: mantenere tutti i componenti del sistema attivi.

I processi alternano fasi di elaborazione CPU-intense con richieste di I/O. Quando la CPU copia/preleva dati dalla RAM si parla di elaborazione e non di I/O.

Tipologie di Processi

1. **Processi Compute-bound (CPU-bound)**: Burst di CPU lunghi, attese di I/O infrequenti, quindi passano molto tempo nella CPU.
2. **Processi I/O-bound**: Burst di CPU brevi, attese di I/O frequenti. Sono tali a causa della bassa necessità di calcoli, non della durata delle richieste di I/O.

Quando Effettuare lo Scheduling

- **Creazione Nuovo Processo**: Decidere quale processo tra figlio e padre scegliere per l'esecuzione.
- **Uscita di un Processo**: Scegliere un altro processo dai pronti o eseguire un processo inattivo del sistema.
- **Blocco di un Processo**: Se un processo si blocca, occorre selezionarne un altro.
- **Interrupt di I/O**: Decidere se eseguire il processo che è appena diventato pronto, quello che era in esecuzione al momento dell'interrupt o qualche altro processo.

Tipologie di Algoritmi di Scheduling e Prelazione

Non Preemptive (Senza Prelazione)

Sceglie un processo e lo lascia eseguire fino a quando si blocca o rilascia volontariamente la CPU. Anche se eseguito per ore, non sarà sospeso forzatamente.

Preemptive (Con Prelazione)

Sceglie un processo e lo lascia girare per un tempo massimo prefissato. Se alla fine dell'intervallo di tempo è ancora in esecuzione, il processo è sospeso e lo scheduler ne sceglie un altro da eseguire. Richiede un interrupt del clock alla fine dell'intervallo per restituire il controllo della CPU allo scheduler.

Scheduling nei Sistemi Batch

I sistemi batch sono utilizzati in attività aziendali periodiche come elaborazione di paghe, inventari, ecc. Nei sistemi batch, spesso sono accettabili algoritmi senza prelazione o con prelazione con lunghi periodi per ciascun processo, riducendo gli scambi di processo e migliorando le prestazioni.

Obiettivi

- **Throughput**: massimizzare il numero di job per ora.
- **Tempo di turnaround**: ridurre al minimo il tempo dallo start al termine di un job.
- **Utilizzo della CPU**: mantenere la CPU sempre impegnata.

Algoritmi di Scheduling nei Sistemi Batch

(a) First-Come, First-Served

Algoritmo di scheduling senza prelazione. I processi vengono assegnati alla CPU nell'ordine in cui arrivano. Quando il processo si blocca viene eseguito il successivo e quando entra nello stato di pronto torna in fondo alla coda.
Vantaggi: facile da capire e da implementare, equo in base all'ordine di arrivo.
Svantaggio: tempi di attesa lunghi per processi I/O-bound in presenza di un processo CPU-bound.

(b) Shortest Job First

Algoritmo di scheduling senza prelazione. Lo scheduler preleva il job più breve. Tempi di esecuzione noti in anticipo.
Vantaggio: minimizza il tempo di turnaround medio quando tutti i job sono disponibili contemporaneamente.
Svantaggio: se i job arrivano in momenti diversi, SJF potrebbe non essere ottimale.

(c) Shortest Remaining Time Next

Versione con prelazione dell'algoritmo SJF. Lo scheduler sceglie sempre il processo che impiegherà meno tempo per terminare l'esecuzione. Il tempo di esecuzione di ciascun processo deve essere noto in anticipo. All'arrivo di un nuovo job, il suo tempo totale è confrontato al tempo restante dei processi attuali. Se il nuovo job richiede meno tempo del processo attuale per terminare, il processo attuale viene sospeso ed è avviato il nuovo job.

Scheduling nei Sistemi Interattivi

In un ambiente con utenti interattivi, l'uso della prelazione è essenziale per evitare che un processo monopolizzi la CPU e neghi il servizio agli altri. La prelazione è

necessaria per prevenire questi comportamenti, specialmente nei server che servono molti utenti remoti.

Obiettivi

- **Tempo di risposta:** Risposte rapide alle richieste dell'utente.
- **Adeguatezza:** Soddisfare le aspettative dell'utente in termini di tempi di risposta.

Algoritmi di Scheduling nei Sistemi Interattivi

Round-Robin Scheduling

È uno degli algoritmi di scheduling più vecchi, semplici, equi e utilizzati. Ogni processo riceve un intervallo di tempo detto "quanto" per l'esecuzione. Se il processo non ha terminato l'esecuzione, la CPU viene prelazionata per un altro processo. Il problema principale è la durata del "quanto". Un quanto tra 20 e 50 ms è spesso ragionevole per bilanciare efficienza e reattività.

Priority Scheduling

Ogni processo ha una priorità, e l'esecuzione è consentita al processo eseguibile con la priorità più alta. Lo scheduler può abbassare la priorità del processo in esecuzione ad ogni interrupt per prevenire l'esclusione indefinita di altri processi. Le priorità possono essere assegnate staticamente o dinamicamente.

Shortest Process Next

Questo algoritmo stima il tempo di esecuzione dei processi basandosi sul comportamento passato. È facile da applicare e migliora il tempo medio di risposta.

Guaranteed Scheduling

Ogni processo ottiene una frazione della CPU proporzionale al numero totale di processi. Il processo con il rapporto più basso tra tempo CPU consumato e quello dovuto viene eseguito fino a superare il suo concorrente più vicino.

Lottery Scheduling

I processi ricevono biglietti della lotteria per le risorse del sistema. Un biglietto viene estratto per decidere quale processo ottiene la risorsa. I processi più importanti possono avere biglietti extra per aumentare le loro possibilità di vincita.

Fair-Share Scheduling

Considera la proprietà dei processi, garantendo che ogni utente riceva una frazione predefinita di CPU indipendentemente dal numero di processi posseduti.

Scheduling nei Sistemi Real-Time

Obiettivi

- **Rispetto delle scadenze:** Assicurarsi che i dati vengano elaborati nei tempi previsti.
- **Prevedibilità:** Garantire un funzionamento costante, specialmente in sistemi multimediali.

I sistemi real-time possono essere divisi in due categorie:

- **Hard Real-Time:** Scadenze assolute da rispettare.
- **Soft Real-Time:** Scadenze mancate sono tollerabili in una certa misura.

Scheduling di Thread

I thread possono essere a livello utente o a livello kernel:

- **Thread a livello utente:** Il kernel ignora l'esistenza dei thread, e lo scheduler interno al processo decide quale thread eseguire.
- **Thread a livello kernel:** Il kernel seleziona un thread specifico per l'esecuzione, e lo scambio del thread comporta uno scambio di contesto.