5 - Gestione della memoria

Gestione della Memoria nei Sistemi Operativi

Introduzione

La memoria principale (RAM) è una risorsa fondamentale che va gestita attentamente. Sebbene la RAM cresca rapidamente, i programmi crescono più velocemente. Idealmente, si desidererebbe avere una memoria privata, grande, veloce e persistente (non volatile), ma la tecnologia attuale non permette ancora questo tipo di memoria.

Nel corso degli anni, è stato sviluppato il concetto di gerarchia della memoria:

- **Memoria veloce e costosa:** pochi megabyte
- **Memoria abbastanza veloce e meno costosa:** qualche gigabyte (volatile)
- **Memoria lenta e poco costosa:** qualche terabyte (non volatile)

È compito del sistema operativo astrarre questa gerarchia attraverso il Gestore della Memoria. I suoi compiti includono: tenere traccia della memoria in uso, allocare nuova memoria e deallocare memoria.

Memoria Senza Astrazione

Il modo più semplice per astrarre la memoria è non farlo. In questo caso, il processo utilizza direttamente la memoria fisica, il che funziona se è eseguito un solo programma. Tuttavia, questo modello fallisce con più programmi e può causare danni se un processo accede all'area di memoria del sistema operativo.

Monoprogrammazione

Esistono tre sottomodelli di organizzazione della memoria:

1. Sistema operativo in fondo alla RAM (mainframe e minicomputer).
2. Sistema operativo in ROM (sistemi embedded).

3. Sistema operativo e drivers in RAM + ROM (primi personal computer, BIOS in ROM).

Multiprogrammazione

È possibile eseguire più programmi contemporaneamente senza astrazione della memoria:

- **Swapping:** il sistema operativo salva l'intero contenuto della memoria su memoria non volatile e carica il programma successivo.
- **Naive Approach:** caricamento di più programmi in memoria fisica consecutivamente senza astrazione dell'indirizzo, causando conflitti e crash.

Astrazione della Memoria

Registri Base e Limite

Per permettere a più applicazioni di risiedere in memoria contemporaneamente senza interferenze, si devono risolvere due problemi: protezione e rilocazione.

- **Protezione:** Etichettare pezzi di memoria con una chiave di protezione (IBM 360).

Spazio degli indirizzi: L'insieme di indirizzi che un processo può usare. Ogni processo ha il suo spazio degli indirizzi personale.

- **Registro base:** Contiene l'indirizzo fisico di inizio di un programma.
- **Registro limite:** Contiene la lunghezza del programma.

Ogni volta che un processo fa riferimento alla memoria, l'hardware della CPU aggiunge automaticamente il valore di base all'indirizzo richiesto dal processo. Se l'indirizzo è maggiore o uguale al valore nel registro limite, viene generato un errore e l'accesso viene terminato.

- **Vantaggi:** Offre a ciascun processo uno spazio di indirizzi protetto e separato.
- **Svantaggi:** Necessità di eseguire somme e confronti ad ogni accesso alla memoria, il che può essere lento.

Swapping

La strategia più semplice di gestione della memoria è lo swapping, che consiste nel prelevare ciascun processo nella sua totalità, eseguirlo per un certo tempo e quindi

porlo nuovamente nella memoria non volatile per non occupare memoria. Quando lo swapping crea spazi vuoti nella memoria (frammentazione), è possibile combinarli spostando tutti i processi il più in basso possibile (memory compaction), ma è un'operazione costosa in termini di tempo CPU.

Un'altra strategia è la **memoria virtuale**, che consente ai programmi di essere eseguiti anche quando sono solo parzialmente nella memoria principale.

Allocazione della Memoria

Quanta memoria allocare a un processo utilizzando lo swapping:

- **Dimensione fissa:** Il sistema operativo alloca esattamente il necessario.
- **Segmenti dati crescenti:** Se i segmenti dei dati dei processi possono crescere, può sorgere un problema di allocazione. Una soluzione è allocare memoria extra durante lo swapping o lo spostamento dei processi. Se un processo non può crescere nella memoria e l'area di swap è piena, deve essere sospeso o terminato.

GESTIONE DELLA MEMORIA LIBERA

Quando la memoria è assegnata dinamicamente, il sistema operativo deve gestirla. Ci sono due modi principali per tenere traccia dell'utilizzo della memoria: bitmap e liste.

Bitmap

La memoria è divisa in unità di allocazione, con ogni unità corrispondente a un bit nella bitmap. Un bit è 0 se l'unità è libera e 1 se è utilizzata. La dimensione della bitmap dipende dalla dimensione della memoria e dall'unità di allocazione. La ricerca di blocchi liberi può essere lenta, poiché richiede la scansione della bitmap.

Liste

Un altro metodo è mantenere una lista concatenata di segmenti di memoria allocati e liberi. Ogni voce della lista contiene informazioni come l'indirizzo di inizio, la lunghezza e un puntatore alla voce successiva. Le liste possono essere implementate come liste singole o doppie, con queste ultime che facilitano la gestione degli spazi liberi contigui.

SCHEMI DI ALLOCAZIONE DELLA MEMORIA

Schemi Comuni

1. **First Fit**: Scansiona la lista dei segmenti finché non trova uno spazio libero abbastanza grande. È veloce ma può causare frammentazione.
2. **Next Fit**: Simile al First Fit, ma inizia la ricerca dall'ultima posizione trovata anziché dall'inizio.
3. **Best Fit**: Cerca lo spazio più piccolo che possa contenere il processo richiesto. Riduce la frammentazione ma richiede più tempo di ricerca.
4. **Worst Fit**: Usa lo spazio più grande disponibile che soddisfa le esigenze del processo.
5. **Quick Fit**: Utilizza liste separate per dimensioni comuni di allocazione, rendendo la ricerca rapida per le dimensioni frequenti. Tuttavia, può causare frammentazione se non gestito adeguatamente.
6. **Buddy Allocation**: Divide la memoria in blocchi di dimensioni potenza di 2. Facilita la coalescenza (unione di blocchi adiacenti) ma può portare a frammentazione interna.

Buddy Allocation (Allocatore Buddy)

Questo metodo inizia con la memoria come un unico blocco contiguo e lo divide ad ogni richiesta in blocchi di dimensioni potenza di 2. Quando un blocco viene rilasciato, cerca di unirlo con il suo "buddy" (blocco adiacente) per formare un blocco più grande. Questo processo può mitigare la frammentazione ma può avere frammentazione interna se un processo richiede un blocco più grande di quello disponibile.

Questi schemi e tecniche sono fondamentali per gestire dinamicamente l'allocazione e il rilascio della memoria nel sistema operativo, garantendo un utilizzo efficiente delle risorse disponibili.