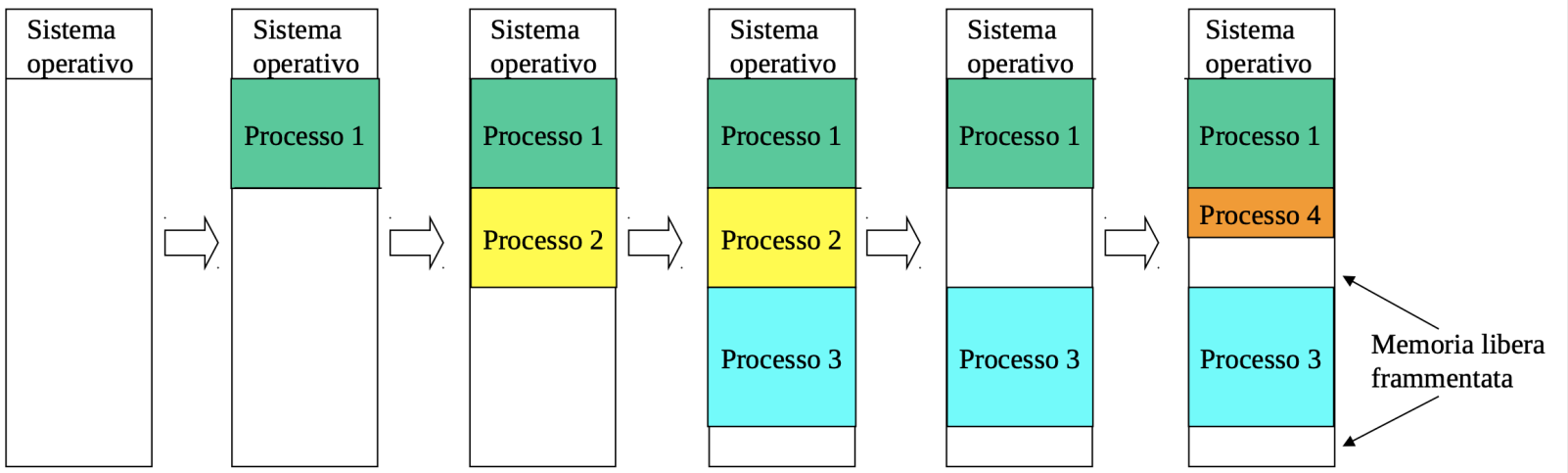
Partizioni dinamiche

giovedì 11 maggio 2023 14:44

Partizioni dinamiche

Qualcosa successivamente è avvenuto rispetto al partizionamento statico e che ha portato ad essere il sistema operativo ad essere più flessibile nell'assegnazione delle ZONE della RAM alle applicazioni che venivano essere attivate.

- si usano partizioni in numero e lunghezza variabile
- quando un processo è caricato in memoria principale, gli viene allocata tanta memoria pari alla sua taglia e mai di più
- la memoria allocata per un processo costituisce una nuova partizione
- quando un processo libera la memoria, una partizione di taglia pari a quella del processo viene resa nuovamente disponibile

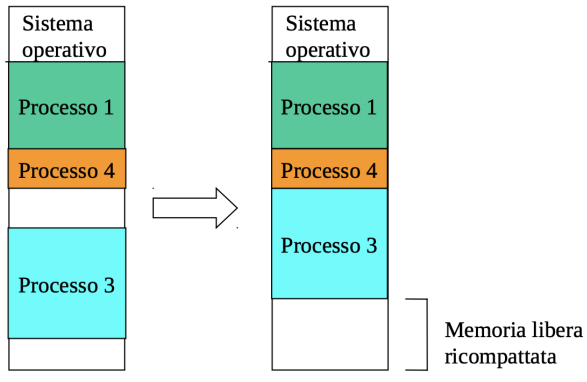


Ogni volta che attiviamo un processo, per quel processo manteniamo la BASE di dove lui è collocato in memoria, e il LIMITE. Quindi se per la causa di un bug esprimiamo la volontà di accedere ad un indirizzo all'esterno della zona del processo stesso, abbiamo una TRAP al sistema operativo che prende il controllo.
Questa cosa la evitiamo con una MMU.
Abbiamo un problema relativo alla FRAMMENTAZIONE ESTERNA DI CUI ABBIAMO PARLATO PRIMA.
Magari la memoria libera frammentata è anche tanta, ma è sparsa qua e là.
Per evitare di INUTILIZZARE QUESTE ZONE perché magari sono troppo piccole per poter lanciare nuovi processi, i cui address space non entrano né sulla prima libera e né sulla seconda, ma magari potrebbero entrare in una quantità di memoria libera che è pari alla somma delle due, si eseguiva la "RICOMPATTAZIONE".

Frammentazione esterna e ricompattazione

La ricompattazione è lo spostamento dei processi attivi all'interno della memoria (il processo 3 viene spostato all'interno di un'altra zona delle memoria fisica) per far sì che le zone libere compaiano come "ricompattate" in un'unica partizione libera di dimensione ampia, da cui possiamo ripartire per allocare nuovi processi.
La ricompattazione è banale nel momento in cui abbiamo il binding a tempo d'esecuzione, perché tanto questo processo ha un codice la cui struttura e i riferimenti che vengono ad essere utilizzati, sono solo logici.
Il fatto che il processo 3 venga spostato e che quindi venga cambiata la base e il limite da un'altra parte, non cambia niente alle attività che sono in esercizio all'interno del processo 3.

- il partizionamento dinamico è soggetto a frammentazione esterna (la memoria esterna alle partizioni diviene sempre più frammentata, e quindi meno utilizzabile)
- per ovviare a tale problema il sistema operativo sposta periodicamente i processi nella memoria in modo da ricompattare le aree di memoria libere



Vincolo
- supporti per il binding a tempo di esecuzione

Svantaggi
- costo elevato

Questo tipo di operazione di ricompattazione ha comunque un costo, perché per spostare le informazioni che sono all'interno della RAM byte per byte da un'altra parte della RAM, impiega numerose istruzioni macchina per eseguire questa ricompattazione.
Ovviamente poi sono state introdotte alcune tecniche particolari che ci permettono di fare questa cosa qua, ossia di andare a far sì che se ho la mia evoluzione dei tempi e all'istante T0 incominciamo a gestire dei processi e dopo un nasce->termina continuo, la frammentazione comincia a prendere piede, quindi ad un certo punto T1 dovevamo andare ad eseguire questa ricompattazione.
C'erano delle tecniche che permettevano di ritardare T1, e quindi pagare MENO FREQUENTEMENTE il costo di dover andare a ricompattare la memoria.
Tutto questo è legato a "come" venivano scelti questi buchi di memoria dopo che veniva richiesto il lancio di un processo, SUPPONENDO CHE PERÒ IL PROCESSO POSSA ENTRARE IN TUTTE E DUE.
Quindi abbiamo i due frammenti ma supponendo che entrambi sono sufficientemente buoni da accogliere il processo, in quale dei due il processo va?

Per questo motivo sono stati introdotti degli algoritmi di allocazione.

Sicuramente è meglio della pagina dell'allocazione dei processi. Erano schemi statici su cui avevamo già definito il numero delle partizioni (incluso il numero massimo) che volevamo ospitare.
Su tutto questo, l'impatto del binding a tempo d'esecuzione è importante soprattutto quando noi abbiamo la necessità di prendere il processo 3 per farlo swappare fuori e liberare la partizione della RAM. Poi verrà riswappato dentro chissà dove. Questo doveva necessariamente impiegare binding a tempo d'esecuzione per quanto riguarda la collocazione di questa applicazione all'interno della RAM.