SISTEMI OPERATIVI

Testi di riferimento:

Dat@Game Hoepli - P.Camagni, R. Nikolassy InfoChef Hoepli - P.Camagni, R. Nikolassy

IL 2° LIVELLO DEL SISTEMA OPERATIVO: IL GESTORE DELLA MEMORIA CENTRALE

Per poter eseguire più processi contemporaneamente indipendentemente dal numero di processori non basta la sola virtualizzazione della CPU, c'è anche bisogno della virtualizzazione della memoria. Questo accade poichè è necessario che questi processi, per poter essere eseguiti, risiedano in memoria centrale. Però N processi insieme generalmente occupano molto più spazio di quello disponibile nella memoria centrale.

Per poter eseguire più processi contemporaneamente indipendentemente dal numero di processori non basta la sola virtualizzazione della CPU, c'è anche bisogno della virtualizzazione della memoria. Questo accade poichè è necessario che questi processi, per poter essere eseguiti, risiedano in memoria centrale. Però N processi insieme generalmente occupano molto più spazio di quello disponibile nella memoria centrale.

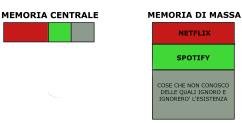
E' COME CERCARE DI INFILARE UN ELEFANTE IN UN FRIGORIFERO!

Bisogna quindi suddividere la memoria centrale tra gli N processi dando ad ogni processo l'impressione di poter disporre dell'intera memoria fisica (**trasparenza**). Come?!

Bisogna quindi suddividere la memoria centrale tra gli N processi dando ad ogni processo l'impressione di poter disporre dell'intera memoria fisica (trasparenza). Come?!

SEGMENTAZIONE

Ogni processo viene suddiviso e caricato in memoria a **segmenti**, ovvero viene caricata in una **porzione continua** di spazio in memoria solo una porzione del programma di dimensione non necessariamente prefissata.



Ad un processo infatti generalmente durante la sua esecuzione non serve avere a disposizione tutte le istruzioni del programma che sta eseguendo, ma solo una porzione di quelle struzioni. Ad esempio: se sto ascoltando la traccia:"le tagliatelle di Nonna Pina" è inutile caricare in memoria tutte le tracce dello Zecchino d'Oro dal 24 settembre 1959 ad oggi... mi basterà caricare in memoria solo quella.

Ogni processo quindi avendo caricato il proprio segmento in memoria pensa di poter sfruttare tutta la memoria liberamente, in verità però questa è solo un'illusione in quanto il Sistema Operativo ogni volta che il processo richiede di utilizzare segmenti nuovi per proseguire procederà allo **swapping**, ovvero al caricamento/scaricamento della memoria per consentire al processo di continuare la sua esecuzione senza che esso riesca a rendersi conto di non star sfruttando tutta la memoria ma solo parte di essa.

Questa tecnica però genera il

PROBLEMA DELLA FRAMMENTAZIONE, ovvero: ho una memoria di 10 unità, carico un segmento di un processo A nelle prime 5 unità, un segmento di un processo B nelle successive 4 unità e un ultimo segmento di un processo C nell'ultima unità:



Ad un certo punto decido di chiudere il processo A e il processo C e di mandare in esecuzione un nuovo programma D che ha bisgono di 6 spazi di memoria per poter essere eseguito.



Teoricamente ci sarebbero 6 spazi per ospitare il segmento del processo D in memoria, ma siccome essi non sono contigui il segmento che ne richiede 6 contigui non può essere caricato!

Un modo per risolvere questo problema è utilizzando la **Deframmentazione**, che sposta i segmenti in modo da lasciare lo spazio inutilizzato tutto all'inizio o alla fine della memoria. Purtroppo però è un'operazione molto costosa in termini di tempo e consumo delle risorse.



Alla segmentazione si affianca quindi un'altra tecnica: la **PAGINAZIONE**: spezza la memoria logica e fisica in pagine di **dimensione fissa** (non più variabile). In questo modo la frammentazione diminuisce e si ottengono prestazioni migliori.