Cap10 - Memoria virtuale parte 1

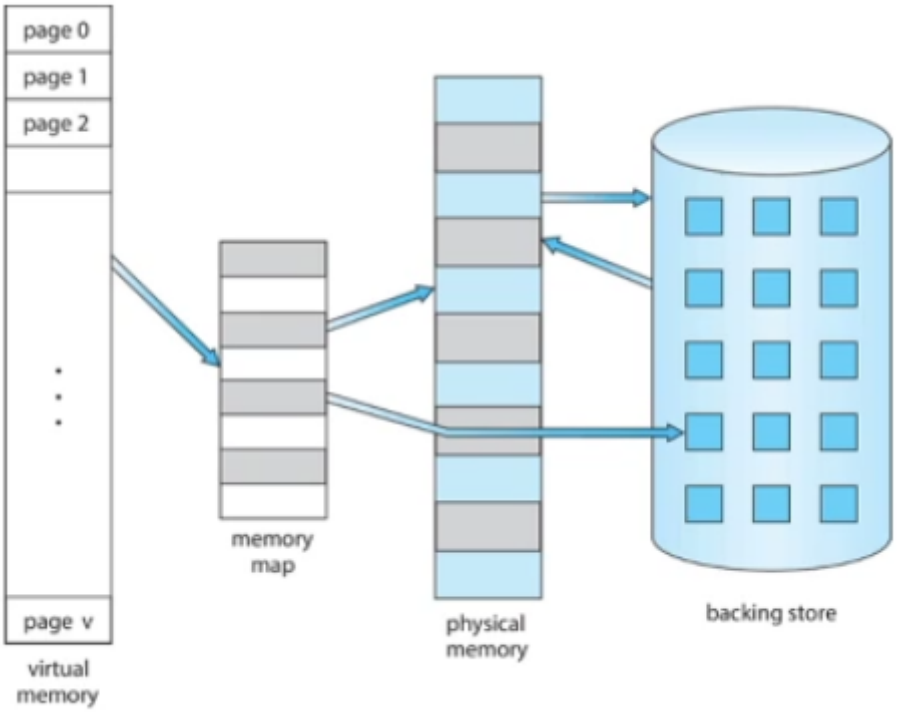
La memoria virtuale è un aspetto particolare della cosidetta paginazione a richiesta.

Essa ha come obiettivo la definizione di uno spazio di memoria di indirizzamento virtuale che sia mappato solo parzialmente in memoria in modo dinamico ("a richiesta").

Ricordiamo che non è necessario che tutto un programma si trovi in memoria nello stesso istante di tempo. Deve esistere un supporto per fare eseguire un programma parzialmente in memoria.

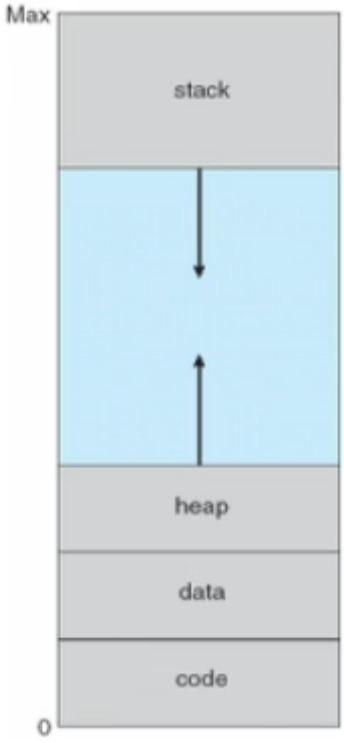
Memoria virtuale: separazione tra la memoria logica e quella fisica. Una parte degli indirizzi logici sono mappati ad indirizzi fisici ed una parte no. Questo ha come conseguenza il fatto che, se la RAM ha una certa dimensione, lo spazio di indirizzamento logico potrebbe essere maggiore della dimensione della RAM.

Spazio di indirizzamento virtuale: insieme degli indirizzi che un programma in esecuzione vede. Di solito parte dall'indirizzo 0 ed è uno spazio di indirizzamento contiguo. Gli indirizzi fisici sono invece realizzati mediante page frames. La tecnica principale della gestione della memoria virtuale è, come detto, la paginazione a richiesta.

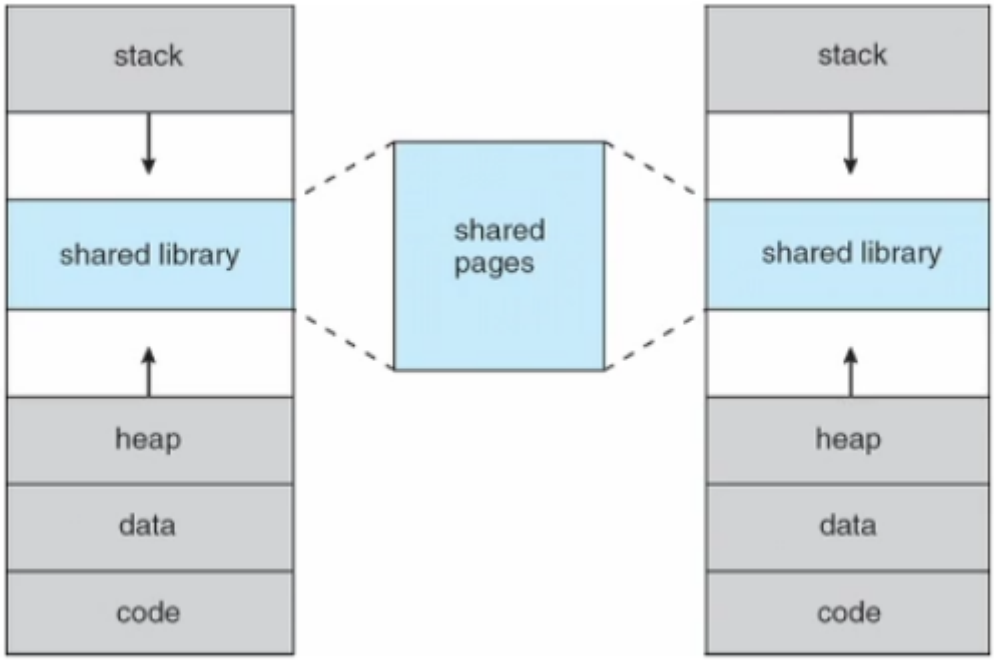


Alcune delle pagine possono essere indirizzate direttamente alla memoria fisica, mentre altre sono indirizzate al backing store. E' inoltre possibile mandare pagine da memoria RAM al disco e viceversa.

Spazio di indirizzamento virtuale



Non tutte le pagine devono stare in un frame. E' possibile avere uno spazio di indirizzamento virtuale "non denso", "sparso". L'heap e lo stack possono essere delle aree caratterizzate da una crescita/decrescita dinamica.

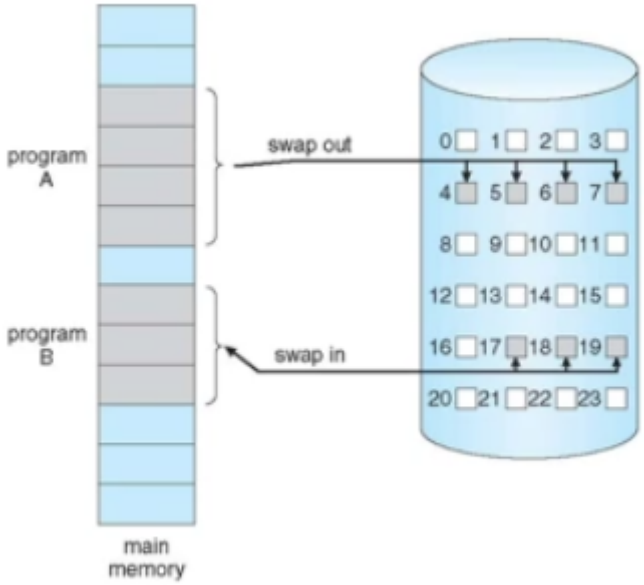


Paginazione a richiesta

Una pagina entra in un frame solo quando essa è richiesta. In tal modo:

- Meno I/O necessario, nessun I/O non necessario
- Meno memoria necessaria
- Risposta più rapida
- Più utenti

La paginazione a richiesta è molto simile al sistema di paginazione con swapping.

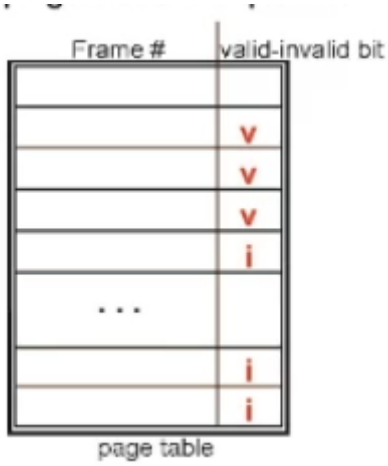


Si effettua inoltre il lazy swapping (una sotrta di previsione): non viene effettuato mai lo swapping di una pagina in memoria a meno che la pagina sia necessaria.

Questi processi di caricamento parziale in memoria si basano sulla capacità di indovinare quali pagine saranno necessarie in futuro (Lo stesso concetto del caching). Per effettuare swap e paging, è necessario inoltre il supporto della MMU.

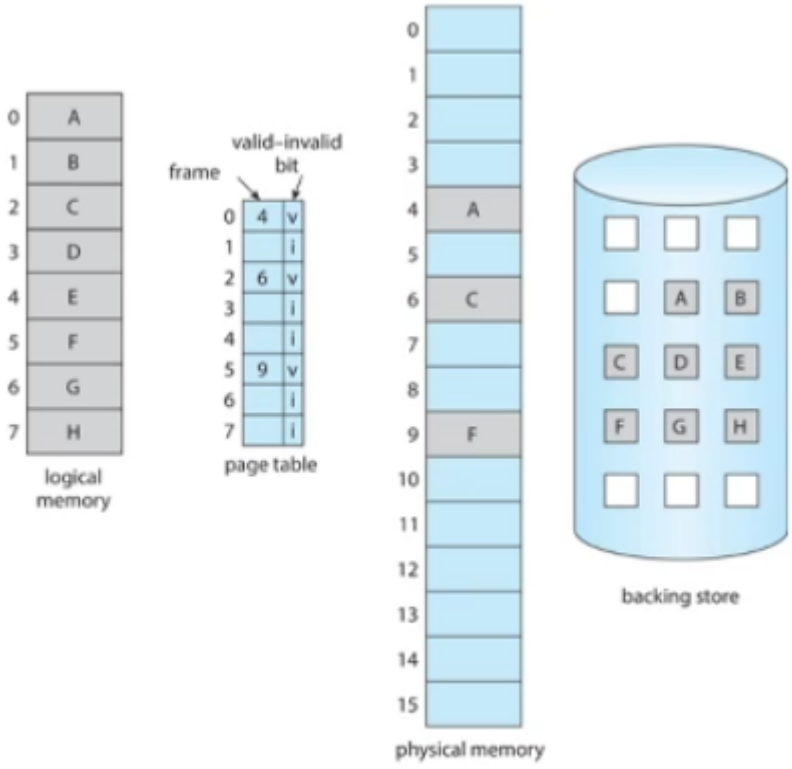
Se la pagina necessaria è già in memoria, non vi sono differenze con la pagina non a richiesta. Altrimenti, essa deve essere caricata in memoria dallo storage.

Nella page table (o nella TBL), di fianco al numero di frame c'è un bit di validità/non validità.



Esso serve ad indicare se la pagina sta in un frame.

Supponiamo di effettuare un accesso in una tabella delle pagine.

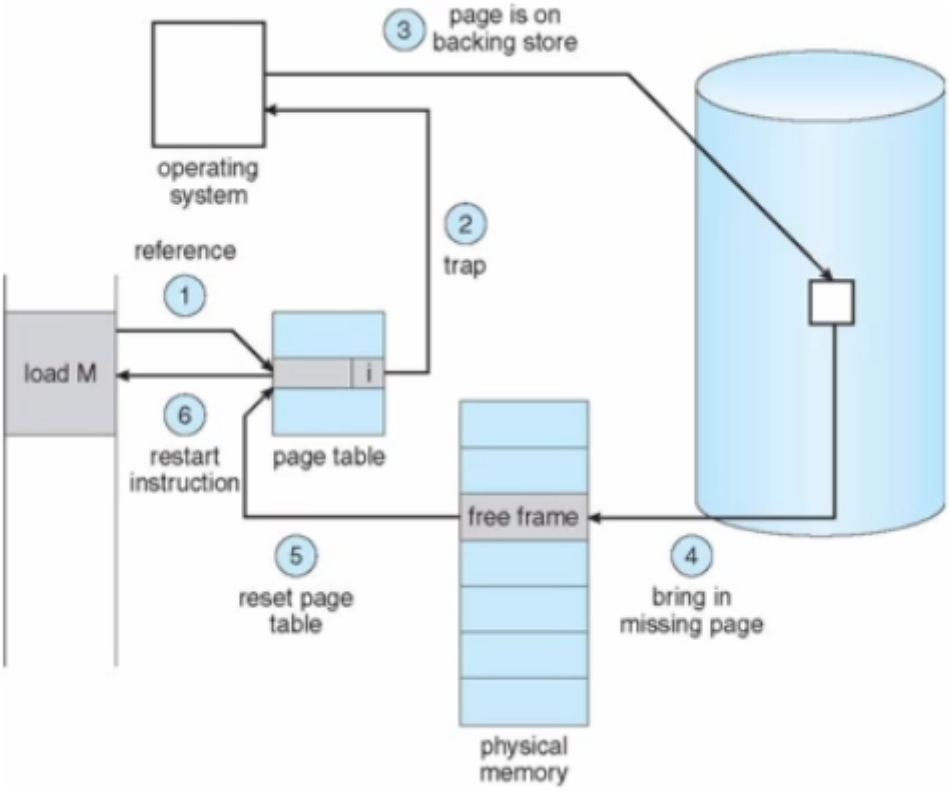


Il bit di validità di A-C-F sono posti a valid, mentre gli altri a invalid. Inoltre, è importante notare che nel backing store possono essere presenti anche delle copie delle pagine che sono presenti nel frame (magari da aggiornare).

Page fault

1. Si parte con un riferimento ad una pagina che non è in un frame (bit di validità = I): viene generato un page fault.
2. L'OS controlla un'altra tabella per verificare se l'invalid reference è un errore oppure se la pagina semplicemente non è in RAM.
3. Se non è in RAM, viene trovato un frame libero.

4. Viene porta la pagina nel frame a partire dal disco.
5. Viene modificata la page table e il bit relativo a tale pagina.
6. Viene riavviata l'istruzione che ha causato il page fault.



Un caso estremo della paginazione a richiesta è un processo che non ha alcuna pagina in memoria. Questo è un caso di **pure demand paging**: l'OS setta l'instruction pointer alla prima istruzione del processo, non residente in memoria. Viene generato un page fault. La gestione del page fault deve potenzialmente portare più pagine in memoria.

Possono esserci inoltre istruzioni che effettuano l'accesso a più location. A volte, per gestire un page fault, è necessario portare in memoria più di una pagina contemporaneamente.

Free-Frame List

La ricerca di un frame libero è necessaria per risolvere un page fault. Un modo per risolvere tale problema agilmente è mediante l'utilizzo di una lista di frame liberi. Il gestore del page fault deve avere un puntatore al primo frame libero per ottenere un frame disponibile. La frame list all'inizio contiene tutta la memoria disponibile.

Worse Case Page Fault

1. Trap (call) al sistema operativo.
2. Fermare il processo e mandarlo in stato di ready. Salvare i registri del processo.
3. Determinare se l'interrupt è legato ad un page fault.
4. Determinare se la page reference è legale e determinare la posizione della pagina sul disco.
5. Ordinare una lettura dal disco a un frame libero:
    1. Aspettare in una coda questo dispositivo finché la richiesta di lettura non viene servita
    2. Aspettare il tempo di ricerca e/o latenza del dispositivo
    3. Iniziare il trasferimento della pagina in un frame libero
6. Nell'attesa, allocare la CPU a qualche altro utente

7. Ricevere un interrupt dal sottosistema I/O del disco (I/O completato)
8. Salvare i registri e lo stato di processo dell'altro utente
9. Determinare che l'interrupt era dal disco
10. Correggere la tabella delle pagine e altre tabelle per mostrare che la pagina è ora in memoria
11. Aspettare che la CPU sia allocata di nuovo a questo processo
12. Ripristinare i registri, lo stato di processo e la nuova tabella delle pagine dell'utente, e poi riprendere l'istruzione interrotta.

Detto ciò, è possibile constatare che un page fault è molto costoso.

Ci sono tre principali attività:

- Servizio di interrupt.
- Lettura della pagina da disco a memoria (I/O). E' il più costoso.
- Far ripartire il processo.

La probabilità di page fault p è compresa fra 0 e 1.

- se p = 0 nessun page fault
- se p = 1, ogni riferimento è un fault

Chiamiamo Effective Access Time (EAT) la seguente formula:

EAT = (1 – p) x accesso alla memoria

+p (overhead per il page fault

+swap out della pagina

+swap in della pagina

Effettuando i calcoli, verrà fuori che il tempo medio di servizio del page fault è di 8 millisecondi (rispetto ai 200 nanosecondi di accesso alla memoria).

In sostanza, l'EAT tenderà a 8 microsecondi. Un rallentamento di un fattore 40!

Se si vuole mantenere una prestazione paragonabile a quella iniziale (con un peggioramento minore del 10%) bisogna avere una probabilità di page fault minore di 2.5*10^-6 (un page fault ogni 400.000 accessi alla memoria).