Problemi di implementazione della memoria virtuale

Le sfide dell'implementazione della memoria virtuale si dividono in:

- Scelta di algoritmi teorici (Es: Seconda Chance, Aging).
- Scelta di pratiche operative (Es: Allocazione globale/locale).
- Gestione di problemi pratici di implementazione della memoria virtuale.

Sistema operativo e paginazione

I momenti in cui il SO svolge delle attività correlate alla paginazione sono durante:

- **Creazione del processo**

> **Attività durante la creazione di un processo**
>
> 1. Determinare le dimensioni iniziali del programma e dei dati.
> 2. Creare ed inizializzare la tabella delle pagine.
> 3. Allocare spazio nella memoria non volatile per lo scambio.
> 4. Inizializzare l'area di scambio e registrare informazione nella tabella dei processi.

- **Esecuzione del processo**

> **Attività durante l'esecuzione del processo**
>
> 1. Azzerare la MMU e, se necessario, svuotare la TLB.
> 2. Rendere attiva la tabella delle pagine, copiandola in un registro specifico.
> 3. Facoltativamente possiamo caricare alcune pagine in memoria per prevenire dei page fault (**Pre-paginazione**).

- **Gestione dei page fault**

> **Attività durante la gestione dei page fault**
>
> 1. Leggere indirizzi hardware per determinare quale indirizzo virtuale ha causato l'errore.

2. Localizzare la pagina necessaria nella memoria non volatile.
3. Trovare un frame disponibile per mettere la nuova pagina, sfrattando altre pagine.
4. Caricare la pagina nel frame, leggerla e ripristinare il **contatore del programma.**

- **Chiusura del processo**

Attività durante la chiusura del processo

1. Rilasciare la tabella del pagine, pagine in memoria e lo spazio in memoria non volatile.
2. Gestire le pagine condivise, le pagine condivise vengono rilasciate solo quando non ci sono più processi che le usano.

Gestione dei page fault

Ecco cosa succede in dettaglio durante un page fault:

1. L'hardware esegue la trap nel kernel, salvando il contatore del programma nello stack, le informazioni sull'istruzione corrente sono salvate su dei registri della CPU.
2. Eseguita una routine in assembly per salvare registri e informazioni volatili, poi chiama il **gestore dei page fault**.
3. Il SO determina la pagina mancante

Dove si trova la pagina mancante

Se non disponibile dai registri hardware, allora bisogna recuperare e analizzare l'istruzione presente nel program counter.

4. Una volta noto l'indirizzo virtuale che ha causato l'errore, il sistema controlla che sia valido e che la protezione sia coerente con l'accesso. Se non lo è viene spedito un segnale e il processo terminato, altrimenti si cerca un frame libero.
5. Se non ci sono frame liberi, si esegue un algoritmo di sostituzione delle pagine e se la pagina è "sporca" viene schedulata per la scrittura in memoria non volatile e il processo sospeso (**Scambio di contesto**).
6. Una volta liberato il frame, il SO schedula un operazione per trovare l'indirizzo contenente la pagina necessaria per poi portarla in memoria, durante il caricamento della pagina viene eseguito un altro processo.

7. Quando l'interrupt del disco indica che è arrivata la pagina, vengono aggiornate le tabelle delle pagine e il frame viene contrassegnato in stato normale.
8. Viene caricata la pagina nel frame e l'istruzione in errore è riportato allo stato iniziale e il contatore ripristinato per puntare a quell'istruzione.
9. Il processo in errore è schedulato e il SO torna alla routine che lo aveva invocato.
10. La routine di servizio ricarica i registri e le informazioni di stato, il controllo ritorna allo spazio utente per continuare l'esecuzione da dove era stata interrotta.

Bloccare le pagine in memoria

Memoria virtuale ed I/O interagiscono in maniera contemporanea.

> **Scenario**
>
> Se un processo invia una richiesta di lettura e scrittura da un file o dispositivo in un buffer nel suo spazio di indirizzi, mentre attende il completamento dell'I/O, può essere sospeso per permettere l'esecuzione di un altro processo.
> Il secondo processo genera un page fault.

Il problema di questo scenario è che ci sta un possibilità che la pagina scelta per la sostituzione sia la pagina contenente il buffer di I/O.
Mentre se avviene un trasferimento DMA su quella pagina, la rimozione potrebbe causare scritture errate nei dati.

La soluzione è quella di **bloccare le pagine impegnate nell'I/O**, questo blocco è detto pinning.

> **Un'altra soluzione**
>
> Gestire tutto l'I/O nei buffer del kernel e copiare i dati nelle pagine utente in seguito, sarebbe necessaria una copia supplementare che rallenta tutto.

Memoria secondaria

La domanda che ci poniamo in questo paragrafo è: Dove vengono poste le pagine che sostituiamo?

Abbiamo nella memoria non volatile una partizione speciale o un dispositivo di memorizzazione dal separato dal file system.
La partizione ha un file system particolare che usa i numeri dei blocchi relativi all'inzio della partizione.

Avvio del sistema

La partizione è vuota, è rappresentata da un voce che indica inizio e dimensione.

- **Avvio del primo processo** -> Riservata una parte di dimensione pari a quella del processo.
- **Avvio di altri processi** -> Ad ogni processo è assegnata una parte della partizione di scambio di dimensione uguale alla loro immagine.

La partizione è gestita come una lista di parti libere e quando un processo termina, si libera la parte di memoria non volatile.

Ad ogni processo è associato l'indirizzo in memoria non volatile della sua area di scambio, l'informazione è contenuta nella tabella dei processi.
Il calcolo dell'indirizzo in cui scrivere una pagina è semplice: basta aggiungere l'offset della pagina nel suo spazio virtuale degli indirizzi all'inizio dell'area di scambio.

Inizializzare l'area di scambio

- Copiare l'intera immagine del processo nell'area di scambio.
- Caricare il processo in memoria e paginarlo quando è in uscita.

⚠ Problema

I processi possono aumentare di dimensioni dopo l'avvio, quindi è meglio creare aree di scambio separate per testo, dati e stack.

Un'alternativa per gestire la memoria secondaria è quella di non allocare niente in anticipo e allocare lo spazio in memoria non volatile per ciascuna pagina quando viene scambiata su disco/SSD, e deallocarlo quando viene riportata in memoria.

Il primo modello comporta una paginazione in area di scambio **statica**, perché ogni pagina ha una posizione fissa sul disco. Il secondo è **dinamico**, perché l'indirizzo su disco è scelto al momento dello scambio.

Si possono anche usare dei file pre-allocati nel file system normale per mantenere le pagine, questo modello è usato in Windows.