7 - Memoria virtuale jnj

Memoria Virtuale

La memoria virtuale permette a un computer di compensare la carenza di memoria fisica usando spazio su disco, creando l'illusione di un grande spazio di memoria continuo. Ecco una sintesi del funzionamento e della struttura della memoria virtuale:

Funzionamento della Memoria Virtuale

1. **Indirizzamento Virtuale**: Ogni processo ha un proprio spazio di indirizzamento virtuale, diviso in pagine.
2. **Mapping**: Gli indirizzi virtuali sono mappati agli indirizzi fisici tramite una tabella delle pagine.
3. **Page Faults**: Se una pagina non è in memoria fisica, viene caricata dal disco.

Struttura della Memoria Virtuale (paginazione)

La maggior parte dei sistemi di memoria virtuale utilizza una tecnica chiamata paginazione, che prevede l'uso di unità di memoria di dimensioni fisse chiamate pagine.

1. **Spazio di Indirizzamento Virtuale**:
   - Diviso in pagine di dimensione fissa (es. 4 KB).
   - Ogni processo ha un proprio spazio di indirizzamento isolato.
2. **Page Table**:
   - Ogni processo ha una tabella delle pagine che mappa gli indirizzi virtuali a quelli fisici.
   - Contiene bit di presenza, indirizzo di frame, bit di protezione, bit di riferimento e bit modificato.
3. **Memory Management Unit (MMU)**:
   - Hardware che traduce indirizzi virtuali in fisici usando la tabella delle pagine.
   - Include il Translation Lookaside Buffer (TLB) per velocizzare le traduzioni.

Gestione degli Accessi e Page Fault

- **Bit Presente/Assente**: Tiene traccia di quali pagine sono presenti in memoria fisica.
- **Page Fault**: Se il programma accede a un indirizzo non mappato, la MMU genera una page fault. Il sistema operativo preleva un frame poco utilizzato, scrive il suo contenuto su disco se necessario, carica la pagina richiesta nel frame liberato e aggiorna la mappa della MMU.

Processi di Gestione della Memoria Virtuale

Quando un indirizzo virtuale viene presentato alla MMU, l'indirizzo di 16 bit viene suddiviso in un numero di pagina di 4 bit e un offset di 12 bit. La MMU utilizza il numero di pagina come indice nella tabella delle pagine per trovare il numero di frame corrispondente. Se il bit presente/assente è 0, si genera una page fault. Se il bit è 1, il numero di frame viene combinato con l'offset per ottenere l'indirizzo fisico.

Vantaggi della Memoria Virtuale

- **Isolamento della Memoria**: I processi sono isolati l'uno dall'altro, aumentando la sicurezza e la stabilità del sistema.
- **Utilizzo Efficiente della Memoria**: Permette l'uso di spazi di indirizzamento più grandi della memoria fisica disponibile.
- **Gestione Semplice della Memoria**: Facilita la gestione della memoria condivisa e consente di eseguire applicazioni più grandi della memoria fisica disponibile.

Svantaggi della Memoria Virtuale

- **Overhead di Page Fault**: Il caricamento delle pagine dal disco può essere molto lento rispetto alla velocità della memoria fisica.
- **Complexità del Sistema**: Richiede un'implementazione complessa sia a livello hardware (MMU) che software (gestione del sistema operativo).

In sintesi, la memoria virtuale permette una gestione efficiente della memoria, isolando i processi e supportando programmi più grandi della memoria fisica disponibile.

Translation Lookaside Buffer (TLB)

Funzione della TLB

La TLB è una cache specializzata utilizzata dalla Memory Management Unit (MMU) per migliorare l'efficienza della traduzione degli indirizzi virtuali in indirizzi fisici. Memorizza traduzioni recenti per ridurre il tempo di accesso alla memoria.

Struttura della TLB

1. Entries (Voci)

Ogni voce della TLB contiene:

- **Indirizzo Virtuale**: L'indirizzo virtuale o il numero di pagina virtuale tradotto.
- **Indirizzo Fisico**: L'indirizzo fisico o il numero di frame corrispondente.
- **Bit di Validità (Valid bit)**: Indica se l'entry è valida.
- **Bit di Protezione (Protection bits)**: Specifica i permessi di accesso (lettura, scrittura, esecuzione).

2. Dimensione della TLB

La TLB è piccola rispetto alla tabella delle pagine, con un numero limitato di voci (tra 32 e 512), ma è molto veloce grazie alla sua implementazione hardware.

3. Associazioni

- **TLB Diretta**: Ogni indirizzo virtuale mappato a una singola voce.
- **TLB Associativa**: Qualsiasi indirizzo virtuale mappato a qualsiasi voce.
- **TLB Set-Associativa**: Indirizzi virtuali mappati a un piccolo set di voci.

4. Operazioni della TLB

- **Lookup**: La MMU cerca nella TLB per una voce corrispondente. In caso di TLB hit, si usa l'indirizzo fisico. In caso di TLB miss, si cerca nella tabella delle pagine e si aggiorna la TLB.
- **Replacement**: Se la TLB è piena, una voce viene sostituita usando algoritmi come FIFO, LRU o Random.

5. Gestione del Contesto

- **TLB Flush**: La TLB viene svuotata ad ogni cambio di contesto.
- **Context Identifier**: Ogni voce nella TLB è associata a un identificatore di contesto, permettendo di mantenere le voci anche durante i cambi di contesto.

Tipi di TLB Miss

1. **Soft Miss**: La pagina di riferimento è nella memoria, ma non nel TLB. Basta aggiornare il TLB.
2. **Hard Miss**: La pagina richiesta non è in memoria. Serve un accesso al disco o all'SSD per prelevare la pagina.

Vantaggi della TLB

- **Velocità**: Riduce il tempo per la traduzione degli indirizzi, migliorando le prestazioni del sistema.
- **Efficienza**: Diminuisce il numero di accessi alla tabella delle pagine, riducendo il carico sulla memoria.

Svantaggi della TLB

- **Complessità**: Richiede hardware specializzato e complesso.
- **Dimensione Limitata**: Può contenere solo un numero limitato di voci.

In sintesi, la TLB è essenziale per l'efficienza dei sistemi di memoria virtuale, migliorando le prestazioni del sistema grazie a una rapida cache per le traduzioni degli indirizzi.

Allocazione di Memoria Globale vs. Allocazione Locale di Memoria

Allocazione di Memoria Globale

Definizione

La memoria è gestita a livello globale per tutti i processi, con risorse allocate in base alle esigenze e disponibilità complessiva.

Caratteristiche

- **Condivisione della Memoria**: Risorsa condivisa tra tutti i processi.
- **Flessibilità**: Allocazione dinamica in base alle necessità.
- **Prestazioni**: Ottimizza l'uso della memoria totale, ma può causare contesa.
- **Page Replacement**: Algoritmi considerano tutta la memoria disponibile.

Vantaggi

- **Efficiente Utilizzo della Memoria**: La memoria inutilizzata può essere redistribuita.
- **Adattabilità**: Si adatta dinamicamente alle esigenze dei processi.

Svantaggi

- **Contesa delle Risorse**: Possibile contesa tra processi.
- **Complessità di Gestione**: Richiede algoritmi complessi.

Allocazione Locale di Memoria

Definizione

Ogni processo ha una porzione fissa di memoria, non condivisa con altri processi.

Caratteristiche

- **Isolamento della Memoria**: Ogni processo ha la propria memoria isolata.
- **Limitazioni Fisse**: Quantità fissa di memoria per processo.
- **Prestazioni**: Evita la contesa, ma può essere inefficiente.
- **Page Replacement**: Algoritmi considerano solo la memoria del processo corrente.

Vantaggi

- **Isolamento e Sicurezza**: Migliora sicurezza e stabilità.
- **Semplicità di Gestione**: Ogni processo gestisce solo la propria memoria.

Svantaggi

- **Inefficienza**: Possibile spreco di memoria non utilizzata.
- **Rigidità**: Meno flessibile per esigenze dinamiche.

Confronto Riassuntivo

Caratteristica	Allocazione Globale	Allocazione Locale
Gestione della Memoria	Condivisa tra processi	Riservata per ciascun processo
Flessibilità	Alta	Bassa
Contesa delle Risorse	Potenziale contesa	Minima
Utilizzo della Memoria	Potenzialmente più efficiente	Potenzialmente meno efficiente
Isolamento	Minore	Maggiore
Algoritmi di Sostituzione	Memoria globale	Memoria del processo corrente

In sintesi, l'allocazione di memoria globale offre flessibilità e efficienza, ma può portare a contese. L'allocazione locale offre isolamento e semplicità, ma può risultare inefficiente.

Segmentazione

Definizione

La segmentazione è una tecnica di gestione della memoria utilizzata nei sistemi operativi per suddividere lo spazio di indirizzamento di un processo in segmenti di dimensioni variabili. Ogni segmento rappresenta una diversa unità logica del programma, come un modulo di codice, una tabella di dati o uno stack.

Caratteristiche della Segmentazione

- **Unità Logiche**: Ogni segmento rappresenta una parte logica distinta del programma, facilitando la protezione e la condivisione della memoria.
- **Dimensioni Variabili**: A differenza della paginazione, i segmenti possono avere dimensioni diverse in base alle necessità del programma.
- **Indirizzamento**: Gli indirizzi di memoria in un sistema segmentato sono costituiti da una coppia (numero di segmento, offset), dove il numero di

segmento identifica il segmento specifico e l'offset specifica la posizione all'interno di quel segmento.

Vantaggi della Segmentazione

- **Protezione**: Ogni segmento può avere i propri permessi di accesso, migliorando la sicurezza del sistema.
- **Condivisione**: I segmenti possono essere condivisi tra diversi processi, permettendo la condivisione di librerie di codice.
- **Riflettività Logica**: La segmentazione riflette più da vicino la struttura logica dei programmi rispetto alla paginazione.

Svantaggi della Segmentazione

- **Frammentazione Esterna**: Con il tempo, la memoria può diventare frammentata, lasciando piccoli buchi tra i segmenti che non possono essere utilizzati efficacemente.
- **Complessità di Gestione**: La gestione della memoria segmentata è più complessa rispetto ad altre tecniche come la paginazione.

Struttura della Segmentazione

1. Tabella dei Segmenti (Segment Table)

Ogni processo ha una tabella dei segmenti che contiene le informazioni sui suoi segmenti. Ogni voce della tabella include:

- **Base**: L'indirizzo di partenza del segmento in memoria fisica.
- **Limite**: La lunghezza del segmento.

2. Registro del Segmento

Il sistema operativo utilizza registri specifici per memorizzare le informazioni di base e limite dei segmenti attualmente in uso. Quando un processo viene eseguito, i registri del segmento vengono caricati con i valori appropriati dalla tabella dei segmenti del processo.

3. Conversione degli Indirizzi

Quando un processo accede a un indirizzo di memoria, il sistema operativo utilizza la tabella dei segmenti per convertire l'indirizzo logico (numero di segmento, offset) in un indirizzo fisico:

- **Numero di Segmento**: Utilizzato per individuare la voce appropriata nella tabella dei segmenti.
- **Offset**: Aggiunto alla base del segmento per ottenere l'indirizzo fisico.

Esempio di Utilizzo della Segmentazione

Supponiamo che un processo abbia tre segmenti: un segmento di codice, un segmento di dati e un segmento di stack. La tabella dei segmenti potrebbe apparire come segue:

Numero di Segmento	Base	Limite
0	1000	400
1	2000	300
2	3000	500

Per accedere a un indirizzo logico (1, 150), il sistema operativo:

1. Trova la voce nella tabella dei segmenti con il numero di segmento 1.
2. Aggiunge l'offset 150 alla base 2000 del segmento.
3. Ottiene l'indirizzo fisico 2150.

Confronto con la Paginazione

Caratteristica	Segmentazione	Paginazione
Unità di Allocazione	Segmenti di dimensioni variabili	Pagine di dimensioni fisse
Indirizzamento	Coppia (numero di segmento, offset)	Coppia (numero di pagina, offset)
Frammentazione	Esterna	Interna
Protezione e Condivisione	Basata su segmenti	Basata su pagine
Riflettività Logica	Alta, rispecchia la struttura del programma	Bassa, divisione arbitraria in pagine

In sintesi, la segmentazione offre una gestione della memoria che riflette più da vicino la struttura logica dei programmi e permette una maggiore flessibilità e

protezione, ma può introdurre problemi di frammentazione esterna e maggiore complessità nella gestione rispetto alla paginazione.