Memoria Virtuale

- Generalità
- Hardware di supporto
- Scelte di progetto

1

Considerazioni sulla paginazione e segmentazione

- I riferimenti di Memoria sono tradotti dinamicamente in indirizzi fisici a run time
  - Un processo può essere caricato in memoria principale e scaricato in modo da occupare regioni differenti durante l'esecuzione
- Un processo può essere diviso in pezzi (pagine o segmenti) che non necessitano di essere allocati in maniera contigua in memoria
- Ne consegue che non tutte le pagine o i segmenti di un processo devono essere caricati in memoria centrale durante l'esecuzione

2

Esecuzione di un Programma

- Il S.O. porta in memoria principale pochi pezzi del programma
- **Resident set** – porzione del processo presente in memoria principale
- Se il processore incontra un indirizzo logico che non si trova in memoria, genera un'interruzione
- Il S.O. mette il processo nello stato Bloccato e prende il controllo …

3

Esecuzione di un Programma

- Il pezzo del processo che contiene l'indirizzo logico richiesto viene portato in memoria
  - Il S.O. emette una richiesta di lettura da disco
  - Un altro processo viene mandato in esecuzione, mentre è eseguito l'I/O da disco
  - Quando l'I/O da disco è completato, è generato un interrupt di I/O a cui consegue che il S.O. riporta a Ready il processo precedentemente bloccato

4

Vantaggi di questo approccio

- Più processi possono essere mantenuti in memoria principale
  - Si caricano solo alcuni pezzi di ciascun processo
  - Con più processi in memoria, è probabile che più processi saranno nello stato Ready in ogni istante
- Un processo può essere più grande della memoria principale

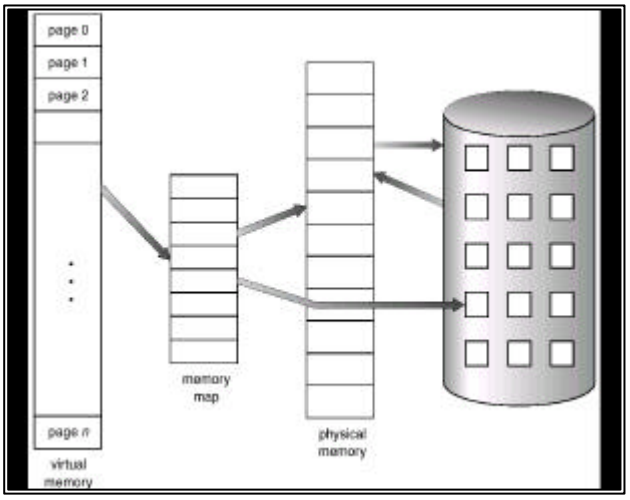
5

Tipi di Memoria

- Memoria Reale
  - Memoria principale
- Memoria Virtuale
  - Memoria su disco
  - Permette una efficace multiprogrammazione e libera l'utente dalle limitazioni della memoria principale

6

La memoria virtuale è più ampia della memoria reale



7

Thrashing

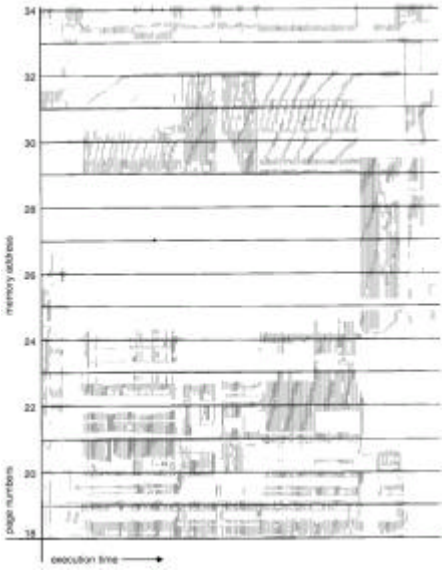
- Può capitare che il S.O. porti fuori dalla memoria un pezzo poco prima che lo stesso pezzo venga richiesto
- Il processore spende la maggior parte del suo tempo a trasferire pezzi piuttosto che a eseguire istruzioni utente:
  - È la condizione di Thrashing!

8

Principio di Località

- I riferimenti al programma o ai dati all'interno di un processo tendono a raggrupparsi
- Solo pochi pezzi di un processo saranno necessari in un breve periodo
- È possibile fare ipotesi su quali pezzi saranno necessari nel prossimo futuro, eliminando il thrashing
- Questo principio suggerisce che la memoria virtuale può lavorare efficientemente

9



10

Supporti necessari per la Memoria Virtuale

- Hardware per supportare la paginazione e/o la segmentazione
- Il S.O. deve gestire i trasferimenti di pagine e/o segmenti tra memoria secondaria e principale

11

Paginazione

- Ogni processo possiede la sua tabella delle pagine
- Ogni entry nella tabella delle pagine contiene il numero di frame della pagina corrispondente in memoria principale
- Un bit è necessario per indicare se la pagina è presente in memoria principale oppure no

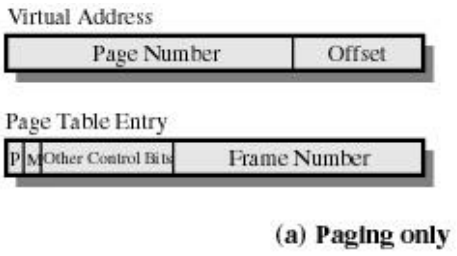
12

Bit di Modifica nella Tabella delle Pagine

- Un bit di modifica serve ad indicare se la pagina è stata modificata da quando è stata caricata in memoria principale
- Se non ci sono stati cambiamenti, la pagina non dovrà essere riscritta sul disco quando dovrà essere scaricata dalla memoria principale

13

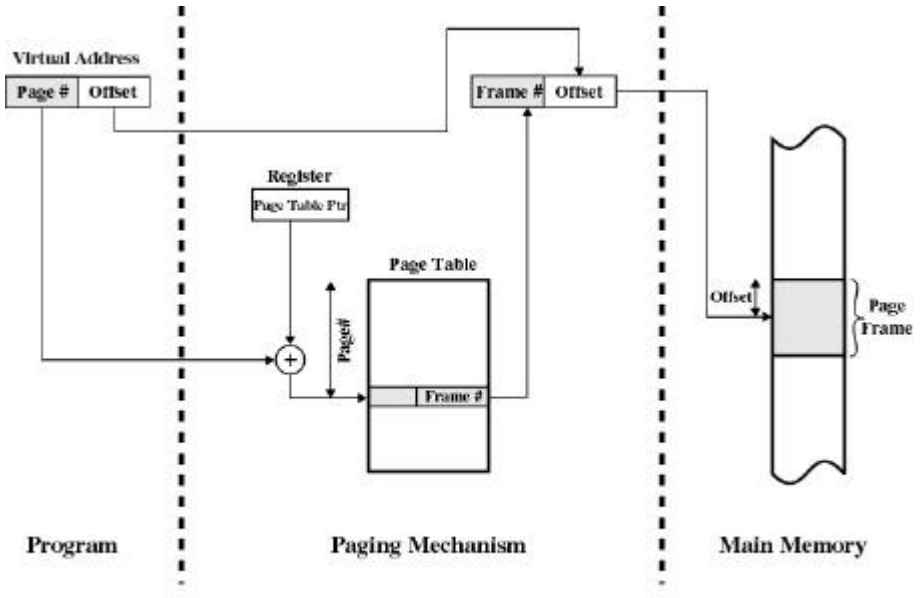
Entry nella Tabella delle Pagine



(a) Paging only

Formato per la gestione della memoria con Paginazione pura

14



Un possibile schema hardware per la traduzione degli indirizzi

Tabelle delle Pagine

- L'intera tabella delle pagine può essere troppo grande ed occupare troppa memoria principale
- Allora, anche le tabelle delle pagine possono essere memorizzate in memoria virtuale
- Quando un processo è in esecuzione, almeno parte della sua tabella delle pagine dovrà essere nella memoria principale.

16

Translation Lookaside Buffer

- Ogni riferimento alla memoria virtuale può causare due accessi alla memoria fisica:
  - Un accesso per caricare la tabella
  - Un accesso per prelevare il dato richiesto
- Per superare questo problema, si può usare una **cache veloce** per le entry della Tabella, detta TLB - Translation Lookaside Buffer
- Il TLB contiene le entries che sono state usate più di recente e funziona come una normale cache di memoria

17

Translation Lookaside Buffer

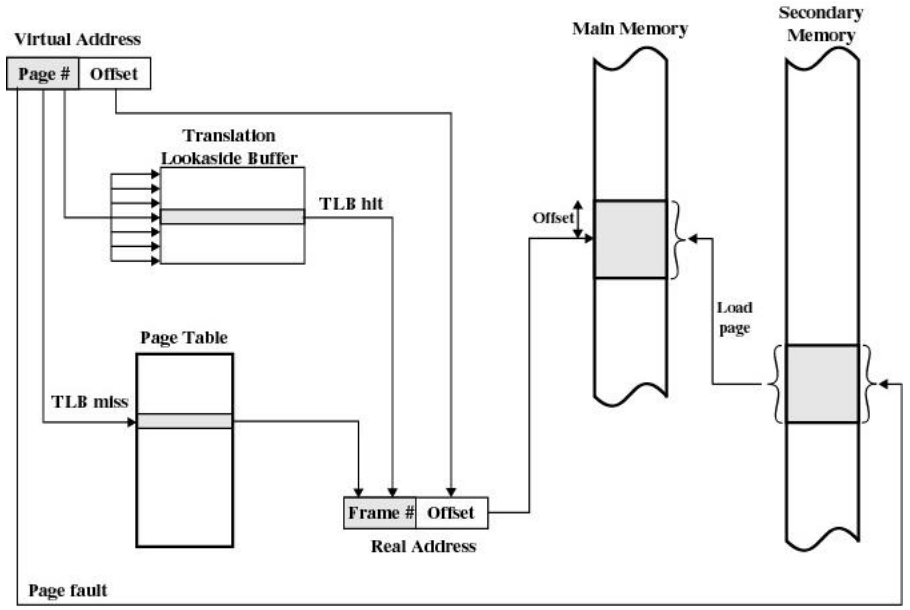
- Dato un indirizzo virtuale, il processore esamina prima il TLB
- Se la entry nella tabella è presente (hit), il numero di frame viene recuperato e si forma l'indirizzo reale
- Se la entry non si trova nel TLB (miss), il processore usa il numero di pagina come indice nella tabella delle pagine, per esaminare la corrispondente entry …

18

Translation Lookaside Buffer

- Se il ‘present bit’ ha valore 1, la pagina è in memoria principale e la entry fornisce il numero di frame
  - Il processore aggiorna il TLB inserendovi la nuova entry
- Se il ‘present bit’ ha valore 0, avviene un page fault
  - Si attiva il S.O. per caricare la pagina e aggiornare la tabella delle pagine

19



Hardware di paginazione con uso del TLB

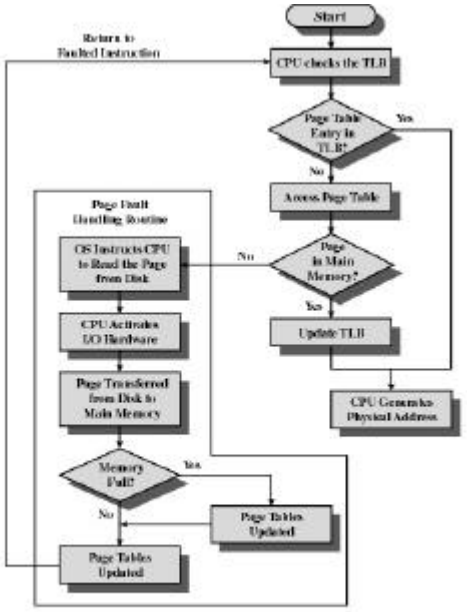


Diagramma di Flusso che mostra l'uso del TLB

Dimensione della Pagina

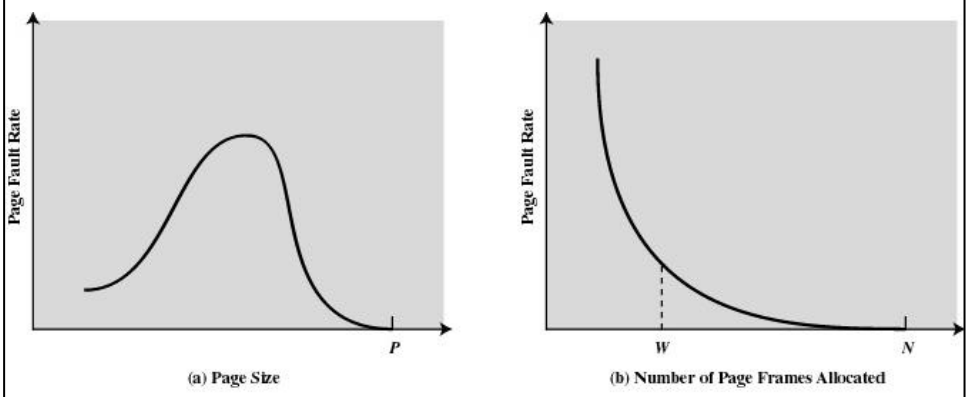
- Minore la dimensione della pagina, minore è la frammentazione interna, però…
- Minore la dimensione, maggiore è il numero di pagine richieste per ciascun processo
- Più pagine per processo significano tabella delle pagine più grande
- Tabelle più grandi implicano ampie porzioni di tabelle in memoria virtuale, con possibilità di doppio page fault
- La memoria secondaria è progettata per trasferire efficientemente ampi blocchi di dati, per cui è preferibile una pagina più grande.

22

Dimensione della Pagina

- Se la pagina è molto piccola, ci sarà un gran numero di pagine per processo presenti in memoria.
- Dopo un certo tempo, le pagine in memoria conterranno porzioni del processo vicine agli accessi recenti. Pertanto, i page fault diminuiscono.
- Al crescere della pagina, ogni pagina conterrà locazioni sempre più lontane dai riferimenti recenti. I page fault aumentano.
- Inoltre, la percentuale di fault dipende anche dal numero di frame allocati per processo…

23



(a) Page Size

(b) Number of Page Frames Allocated

P = size of entire process
W = working set size
N = total number of pages in process

Fissando la dimensione della pagina, la percentuale di page fault diminuisce al crescere del numero di frame allocati

Dimensione della Pagina

- Un'alternativa: l'uso di pagine con dimensioni diverse fornisce la flessibilità necessaria per usare efficacemente il TLB.
  - Pagine grandi per le istruzioni dei programmi (che occupano regioni contigue diindirizzi)
  - Pagine piccole per gli stack dei thread
- La maggior parte dei sistemi operativi supporta una sola dimensione di pagina.

25

Segmentazione

- La segmentazione permette al programmatore di vedere la memoria come insieme di segmenti multipli.
- I segmenti possono avere dimensioni diverse e variabili dinamicamente: i riferimenti a memoria sono indirizzi del tipo (numero di segmento, offset).
- Vantaggi:
  - Semplifica la gestione di strutture dati che crescono
  - Si presta alla condivisione tra processi: un programma d'utilità o una tabella dati possono essere allocati in un segmento accessibile a più processi
  - Si presta alla protezione, assegnando privilegi ai segmenti

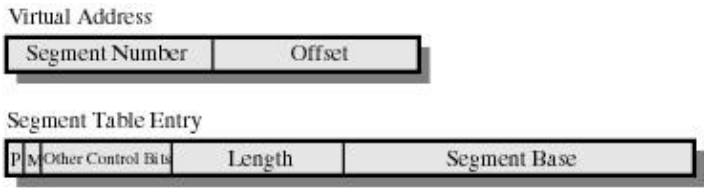
26

Tabelle dei Segmenti

- Simili alle tabelle delle pagine
- Ogni entry contiene
  - Indirizzo di partenza del segmento in memoria e lunghezza del segmento
- Inoltre, per uno schema di memoria virtuale con segmentazione:
  - Un bit per stabilire se il segmento è presente in memoria
  - Un bit di modifica che indica se il segmento è stato alterato dopo l'ultimo caricamento in memoria
  - Altri bit di controllo (es. protezione e condivisione a livello segmento)

27

Entry in Tabella dei Segmenti



(b) Segmentation only

Tipici formati per la gestione della memoria segmentata

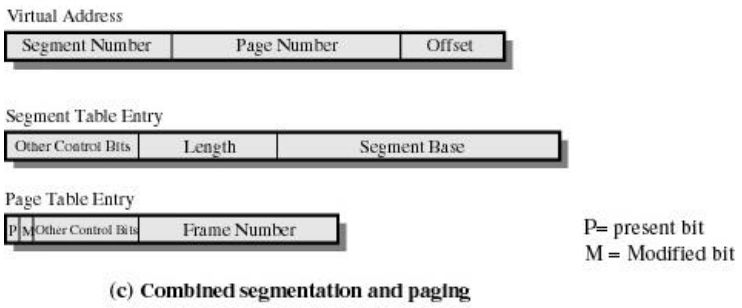
28

Combinare Paginazione e Segmentazione

- La paginazione è trasparente al programmatore ed elimina la frammentazione esterna
- La segmentazione è, invece, visibile al programmatore e consente di gestire strutture dati dinamiche, modularità, e supporto per la condivisione e protezione
- Si possono combinare le due tecniche per ottenere i vantaggi di entrambe
  - Lo spazio di indirizzi utente è ancora diviso in segmenti
  - Ogni segmento è suddiviso in pagine di dimensione fissa (pari a quella del frame di memoria)

29

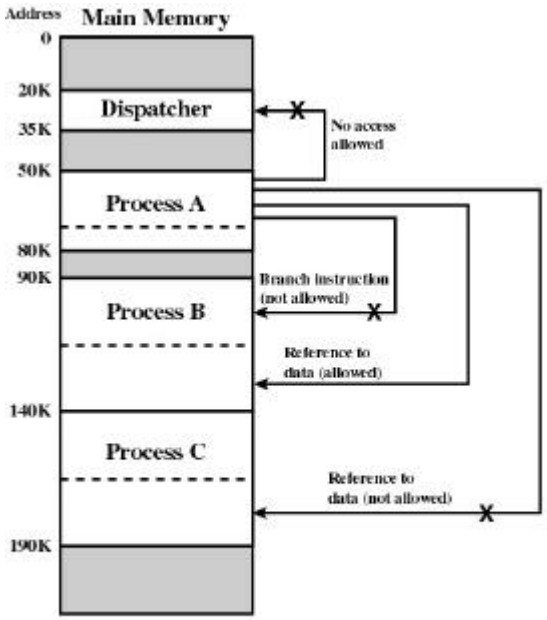
Segmentazione e Paginazione combinate



(c) Combined segmentation and paging

Tipici formati per la gestione di memoria segmentata e paginata

30



Il programmatore può specificare i requisiti di protezione e condivisione per ciascun segmento

Relazioni di protezione tra i segmenti

Le scelte di progetto di un S.O.

- Usare o meno la memoria virtuale
- Paginazione, Segmentazione o entrambe?
- Algoritmi per la gestione della memoria

- Scelte relative alla memoria virtuale (*scopo:minimizzare la frequenza di page fault*):
  - Strategia di Fetch– quando caricare una pagina?
  - Posizionamento – dove caricare?
  - Sostituzione – cosa rimuovere per fare spazio ad una pagina?
  - Gestione del Resident Set– quante pagine caricare?
  - Strategia di Cleaning– quando scrivere una pagina su disco?
  - Controllo del Carico – livello di multiprogrammazione

32

Strategia di Fetch

- Strategia di Fetch
  - Determina quando portare una pagina in memoria
- Paginazione su richiesta (Demand paging)
  - Porta una pagina in memoria solo quando avviene un riferimento ad una locazione della pagina
    - Molti page faults quando un processo viene avviato
- Prepaginazione: porta in memoria più pagine del necessario
  - È più efficiente per caricare pagine che sono memorizzate su aree contigue del disco
  - Ma è difficile indovinare le pagine che saranno necessarie

33

Strategia di Sostituzione

- Strategia di Sostituzione
  - Quale pagina sostituire quando una nuova pagina va caricata?
  - La pagina rimossa dovrebbe essere quella a cui ci si riferirà di meno nel prossimo futuro
  - Per il principio di località, c'è spesso correlazione fra la più recente storia dei riferimenti e le sequenze di riferimenti future…
    - La maggior parte delle strategie cerca di predire il comportamento futuro sulla base di quello passato

34

Strategia di Sostituzione

- Un possibile vincolo: Blocco di Frame
  - Se un frame è bloccato, non può essere sostituito
  - Frame bloccati:
    - Kernel del sistema operativo
    - Strutture di Controllo critiche
    - I/O buffers
  - Si associa un bit di lock ad ogni frame

35

Algoritmi di sostituzione di base

- Algoritmo ottimo
- Least Recently Used (LRU)
- Fist-in, First-out (FIFO)
- Orologio

36

Algoritmi di sostituzione di base

- Algoritmo ottimo
  - Seleziona, per la sostituzione, la pagina alla quale ci si riferirà dopo il tempo più lungo
    - Farà il numero di fault di pagina più piccolo
- Impossibile da implementare
  - Può essere usato solo per giudicare altri algoritmi per confronto con esso

37

Documento Acrobat

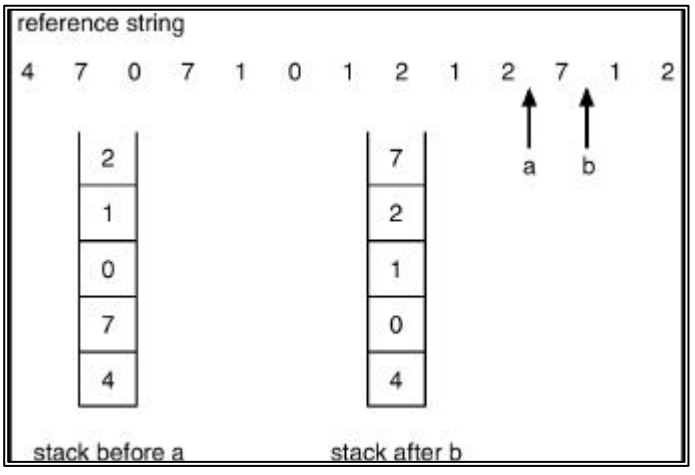
38

Algoritmi di sostituzione di base

- Least Recently Used (LRU)
  - Sostituisce la pagina che non è stata referenziata da più tempo
  - Per il principio di località, questa dovrebbe essere la pagina che meno probabilmente sarà referenziata nel prossimo futuro
- Implementazione
  - ogni pagina dovrebbe essere contrassegnata con l'orario del suo ultimo riferimento. Soluzione costosa dal punto di vista hardware. In alternativa, uso di una pila dei riferimenti di pagina, in cui in cima alla pila c'è sempre la pagina usata per ultima

39

LRU: Uso di una pila per registrare gli ultimi riferimenti alle pagine



40

Algoritmi di sostituzione di base

- First-in, first-out (FIFO)
  - Tratta i frame allocati per le pagine di un processo come un buffer circolare
  - Le pagine sono rimosse usando lo schema round-robin
  - È la più semplice politica da implementare: basta un puntatore che va in cerchio lungo i frame
  - Sostituisce la pagina che è stata più a lungo in memoria (perché potrebbe essere ormai fuori uso).
  - Queste pagine, però, potrebbero essere necessarie durante tutta l'esecuzione (programma o dati molto usati) e dovranno essere ricaricate.

41

Algoritmi di sostituzione di base

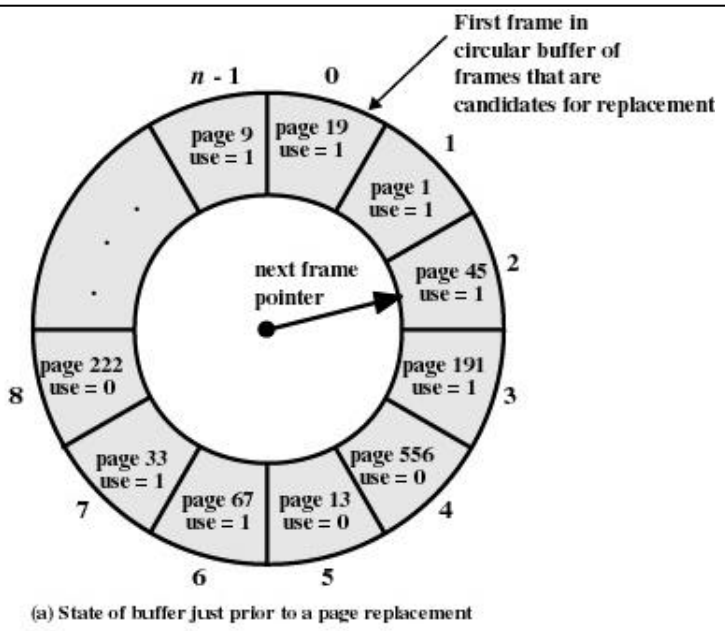
- Algoritmo dell'orologio (*o con seconda chance*)
  - Richiede un bit aggiuntivo per frame, detto **use bit**
  - Quando una pagina è caricata la prima volta in un frame, lo use bit nel frame è fissato a 0. Quando la pagina sarà referenziata successivamente, lo use bit è fissato a 1.
  - L'insieme dei frame candidati alla sostituzione è gestito come un buffer circolare, cui è associato un puntatore.
  - Quando una pagina è sostituita, il puntatore punta al frame successivo del buffer.

42

Algoritmi di sostituzione di base

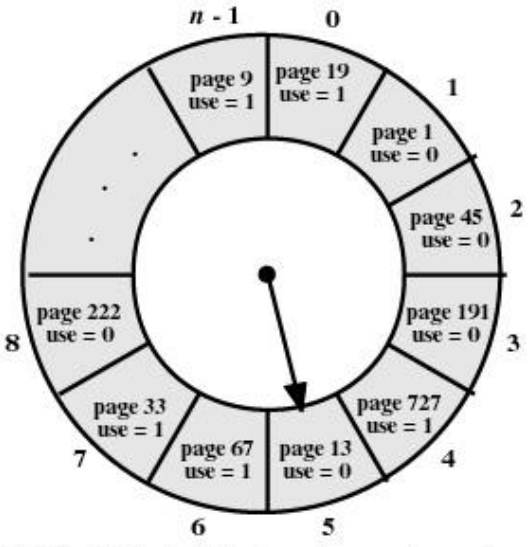
- Quando è tempo di sostituire una pagina, il S.O. scorre il buffer per trovare un frame con use bit a 0.
  - Durante la ricerca per la sostituzione, ogni use bit con valore 1 è cambiato a 0
  - Il primo frame incontrato con lo use bit a 0 viene sostituito.
  - Se tutti i bit hanno valore 1, il puntatore farà un giro completo del buffer, azzerando tutti gli use bit, fermandosi nella posizione originale
  - La strategia è simile alla FIFO, tranne che i frame con use bit a 1 sono evitati dall'algoritmo (viene data loro la 2a chance).
  - Ha prestazioni simili al LRU

43



(a) State of buffer just prior to a page replacement

Figure 8.16 Example of Clock Policy Operation



(b) State of buffer just after the next page replacement

Figure 8.16 Example of Clock Policy Operation

Variante all'algoritmo dell'orologio

- Si considera anche il bit di modifica associato al frame
  - Scandisce il buffer cercando pagine con Mod=0, Use=0: la prima incontrata è selezionata per la sostituzione
  - Se non ne trova, riparte cercando pagine con Use=0, Mod=1, e durante la ricerca azzera lo Use bit di ogni frame incontrato
  - Ripete il passo 1 e poi il 2, se necessario (questa volta troverà un frame per la sostituzione con Use=0)
- Preferisce per la sostituzione pagine non modificate, né usate recentemente, perché non hanno bisogno di essere riscritte su disco (risparmio di tempo).
  - Vedere Figura 8.16, page 379

46

Algoritmi di sostituzione di base

- Uso di Buffer per pagine
  - Si usa l'algoritmo FIFO per la sostituzione, ma…
  - La pagina da sostituire rimane in memoria, e viene solo spostata in una di due liste:
    - Lista delle pagine libere, se la pagina non è stata modificata
    - Lista delle pagine modificate
  - Quando una pagina deve essere caricata, si usa il primo frame della prima lista (sostituendo la pagina corrispondente).
- Vantaggi
  - La pagina da sostituire resta comunque in memoria e la si può rimettere nel resident set rapidamente
  - Le pagine modificate sono scritte tutte insieme piuttosto che una alla volta (riducendo il numero di operazioni di I/O)

47

Dimensione del Resident Set

- Allocazione Fissa
  - Assegna al processo un numero fissato di pagine per l'esecuzione
  - Quando avviene un page fault, una delle pagine del processo deve essere sostituita
- Allocazione Variabile
  - Il numero di pagine allocate al processo varia durante la sua esecuzione (a seconda dei page fault generati)
    - È una strategia più potente ma provoca overhead

48

Ambito di sostituzione

- Ambito di sostituzione locale
  - La scelta della pagina da sostituire avviene tra le pagine in memoria del processo che ha generato il page fault
- Ambito di sostituzione globale
  - Tutte le pagine non bloccate in memoria sono candidate alla sostituzione (indipendentemente dal processo cui appartengono)
  - È una strategia più facile da implementare e con meno overhead

49

Allocazione fissa, ambito locale

- Il S.O. sceglie una pagina da rimpiazzare tra i frame allocati al processo
- Il numero di frame allocati è prefissato
  - Se il numero è troppo piccolo, ci saranno molti page fault
  - Se il numero è troppo grande,ci saranno pochi processi in memoria
    - Il processore lavorerà lentamente oppure passerà molto tempo a trasferire processi sul disco

50

Allocazione variabile, ambito globale

- La più semplice da implementare e adottata da molti S.O.
- Il S.O. tiene la lista dei frame liberi
- Quando avviene un page fault, un frame libero è aggiunto al resident set del processo e vi si carica la pagina.
- Problema della scelta della sostituzione:
  - Se non ci sono frame liberi, sarà sostituita una pagina di un altro processo (che potrebbe averne bisogno)
  - Per evitare questo inconveniente, conviene usare i buffer di pagine

51

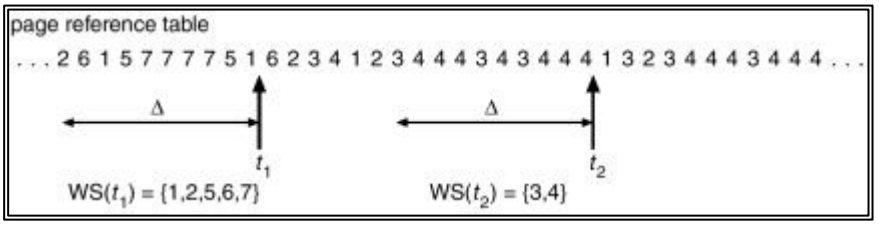
Allocazione variabile, ambito locale

- Cerca di superare i problemi della precedente strategia
- Quando un nuovo processo è caricato, gli si alloca un numero di frame che dipende dal tipo di applicazione, richieste del programma, o altri criteri.
- Quando avviene un page fault, si seleziona la pagina da sostituire tra quelle del resident set del processo che provoca il page fault .
- Periodicamente, si rivaluta l'allocazione fornita al processo e la si adatta, per migliorare le prestazioni globali.

52

Strategia del Working Set

- Il Working Set con parametro Δ per un processo al tempo t , W(t, Δ), è l'insieme di pagine del processo cui ci si è riferiti nelle ultime Δ unità di tempo in cui il processo è stato in esecuzione.
- Δ è l'ampiezza della finestra del working set (es. numero di istruzioni osservate)
- Se una pagina è in uso attivo, si trova nel Working Set
- Se non è più usata, esce dal WS dopo Δ u.tempo



Strategia del Working Set

- La dimensione del W.set in genere può variare nel tempo
- La scelta di Δ influisce sulla precisione del WS:
  - Δ piccolo -> non include la località di P
  - Δ grande ->può sovrapporre più località
  - $\Delta = \infty \Rightarrow$ coincide con tutte le pagine usate durante l'esecuzione di P
- Il S.O. controlla i Working Set dei vari processi, per decidere sul resident set:
  - Rimuove periodicamente le pagine che non sono nel working set del processo
  - Un processo può essere eseguito solo se il suo working set è in memoria principale

54

Strategia del Working Set

- Problemi:
  - Il passato non predice sempre il futuro (WS e la sua dimensione cambiano nel tempo)
  - È difficile misurare il working set per ogni processo
- Un'approssimazione della strategia, piuttosto che guardare alla dimensione del WS guarda alla frequenza di page fault
  - Se la frequenza è minore di una soglia, si rimuovono pagine dal resident set
  - Se la frequenza è maggiore, si aggiungono pagine
  - Può essere necessario sospendere qualche processo

55

Strategia di Cleaning

- Determina quando una pagina modificata deve essere scritta in memoria secondaria
- Cleaning a richiesta: Una pagina è scritta solo quando è stata scelta per essere sostituita
  - La scrittura di una pagina modificata precede la lettura di una nuova pagina (Un processo potrebbe aspettare due trasferimenti su disco per proseguire)
- Precleaning: Scrive le pagine modificate prima che i loro frame siano necessari (vantaggio: le pagine possono essere scritte in gruppi)
  - le pagine scritte restano in memoria principale finchè l'alg. di sostituzione non ne ordina la rimozione
  - Tali pagine potrebbero essere state modificate ancora, prima di essere sostituite

56

Strategia di Cleaning

- Il miglior approccio usa buffer per le pagine
  - Le pagine sostituite sono poste in due liste
    - Modificate e non modificate
  - Le pagine nella lista modificate sono scritte in gruppi periodicamente
  - Le pagine nella lista non modificate o sono richiamate quando vengono referenziate di nuovo, oppure sono perse quando il loro frame è assegnato ad un'altra pagina

57

Controllo del Carico

- Determina il numero di processi che saranno residenti in memoria
- Con pochi processi, tutti potrebbero essere bloccati e molto tempo sarebbe sprecato a fare swapping
- Troppi processi possono dar luogo a thrashing

58

Sospensione dei Processi

- Quando il livello di multiprogrammazione è troppo alto e si verifica il trashing, può essere necessario spostare processi su disco. Criteri di scelta:
- Processo con la più bassa priorità
- Processo che provoca fault
  - Non avendo il suo working set in memoria,si bloccherà comunque
- Ultimo processo attivato
  - È meno probabile che il suo working set sia residente

59

Sospensione dei Processi

- Processo col minore resident set
  - Richiederà meno lavoro per essere ricaricato
- Processo più grande
  - Libera un maggior numero di frame
- Processo con la maggiore finestra di esecuzione rimanente

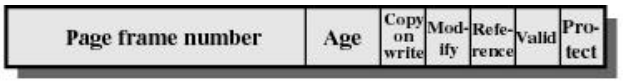
60

UNIX and Solaris Memory Management

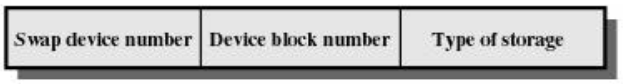
- Paging System
  - Page table
  - Disk block descriptor
  - Page frame data table
  - Swap-use table

61

Data Structures



(a) Page table entry

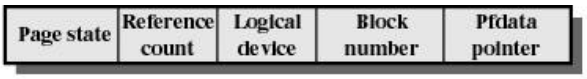


(b) Disk block descriptor

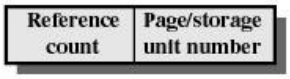
Figure 8.22 UNIX SVR4 Memory Management Formats

62

Data Structures



(c) Page frame data table entry



(d) Swap-use table entry

Figure 8.22 UNIX SVR4 Memory Management Formats

63

UNIX and Solaris Memory Management

- Page Replacement
  - refinement of the clock policy
- Kernel Memory Allocator
  - most blocks are smaller than a typical page size

64