Vantaggi e strategie per migliorare le prestazizoni

Il trashing si può ridurre ed eliminare. Invece non si può eliminare il page fault.

> *Come si elimina il trashing? Come si fa a sapere di quanti frame ha bisogno un processo?*
> Bisognebbe assegnare un numero di frame commisurato alle *"necessità del processo"*:

L'allocazione dei frame aiuta ad allocare i frame ma non dà un numero esplicito minimo di frame necessari per un processo.

Concetto di località

Mentre un processo viene eseguito, esso si sposta da una parte all'altra. Una ***LOCALITA'*** è un **insieme di pagine** che vengono **usate attivamente tutte insieme**. Ne segue che un programma è composto da *differenti località* che pian piano andranno a sovrapporsi.

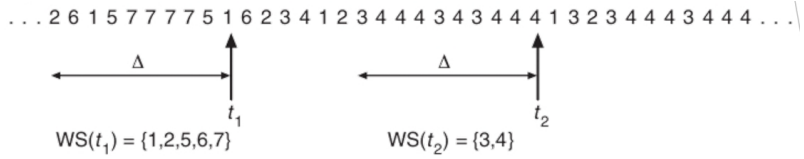
> Programma, struttura delle istruzioni ecc...
> Quando si avvia un processo, lavora su una località.

Definizione: Tutti i programmi mostreranno la struttura di rferimento della memoria di base

- Quando si prende per la prima volta una determinata località, allora in quel momento si va a prendere qualcosa di nuovo che non è ricato in memoria. Quindi, ne segue, che si hanno i page fault
- Alla fine della località avrò anche dei page fault perchè si sta uscendo per avviare qualcosa (una sequenza di istruzioni) di nuovo
- All'interno della località non si dovrebbero avere frequenti page fault

Working Set (WS)

L'idea è quella di osservare con attenzione il numero di ***frame e pagine utilizzate dai processi*** nell'ultimo Δ intervallo di tempo.



- Si identifica, durante l'esecuzione del processo, se vengono referenziate un numero di pagine sufficienti o meno.

> *Il Working Set risiede in memoria e tutte le pagine al suo interno sono pagine che non generano page fault perchè, appunto, sono quelle utilizzate in quell'istante.*

Il *WS* si basa su Delta: Δ = **dimensione della finestra temporale**, quindi contiene il numero di pagine utilizzate in quell'istante

- Ne segue che, se una pagina non è attiva, allora essa non si troverà nel WS
- Δ rappresenta un'approsimazione del concetto di località del processo
- Ci interessa **il numero di frame utilizzati** nel Δ intervallo di tempo

L'**efficienza del *WS*** è determinato dalla **dimensione del Δ:** più è grande Δ, più si ha precisione nell'osservazione del numero di frame usati dal processo. Quindi Δ evidenzia la **precisione**. Per quanto riguarda Δ :

- Se è **troppo piccolo**, non si riesce a capire il numero esatto di frame usati, e si ha una **PICCOLA LOCALITA'**
- Se è **troppo grande**, si ha una cronologia maggiore, ma si avrà comprendere nel WS informazioni dove le **LOCALITA' GRANDI CHE SI SOVRAPPONGONO**

Se $D = \sum WSS_i$ è la dimensione del WS (WWS_i= *Working Set Space*) , si può calcolare la **richiesta totale** di frame che il processo ne ha di bisogno. D deve essere minore della memoria disponibile

Vantaggio nell'uso del *WS*: il *Sistema Operativo* controlla *WS* di ogni processo per capire quanti frame assegnare a ogni singolo processo

Prepaginazione

Se si tiene conto del WS, **dopo** che il processo **si sospende** (*si memorizza l'ultimo WS per quel processo*) e poi deve essere **RIESEGUITO**, si va a **controllare il suo WS** e quindi il numero di pagine usate nell'ultimo Δ e quindi si sa **QUANTE** e **QUALI** pagine ha frequentemente utilizzato

L'operazione che salva l'ultimo WS di un processo dopo essere stato sospeso, permette di ***CARICARE IN ANTICIPO*** l'ultimo *WS* per quel processo e in questo caso si parla di ***PREPAGINAZIONE***. Decisamente si risparmiano i passati iniziali page fault *obbligatori* per ogni processo (*tranne quelli che sono avviati per la prima volta, ovviamente*).

Calcolo WS

- Attraverso **interrupt periodici**: valuta quali pagine sono state chiamate in causa
- **BIT** di referenziamento R
- un **LOG** che conserva la **cronologia** dei referenziamenti legati al Δ:

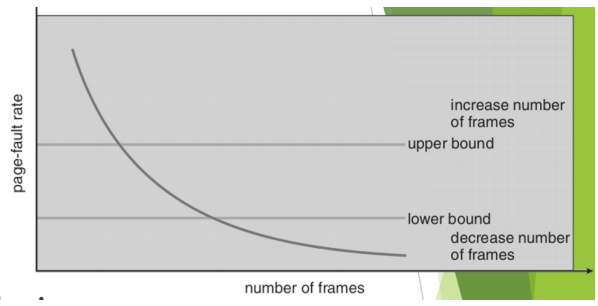
Ci si deve fare un'idea dei frame da assegnare.

Page fault frequency

L'obiettivo principale del sistema operativo è quello di **ridurre** il numero di page fault e, in particolare, **ridurre a 0** il trashing.

> *Il trashing si ha quando è dovuto al numero (*inferiore del minimo*) di frame assegnati a un processo.
> Aumento Memoria -> Diminuisco Page Fault -> Diminuisco Trashing

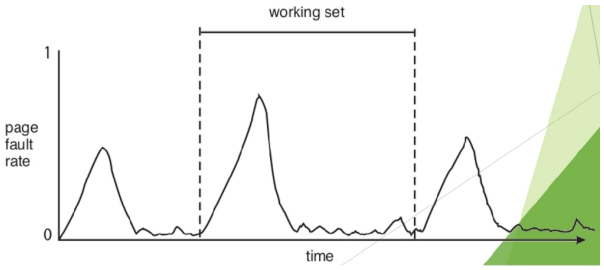
QUOTIDIANAMENTE si controlla il numero di page fault generati da un processo. Si parla allora di ***Page Fault Frequency***.



Si definiscono **Upper Bound** (*UB*) e **Lower Bound** (*LB*) i limiti superiori/inferiori del numero di page fault:

- Se il numero di page fault ricade nell'intervallo (*UB, LB*) vuol dire che si ricade in un **numero normale di page fault**
- Se si **scende sotto la soglia** del *LB* (*inferiore*), allora vuol dire che il numero di page fault è ***BASSO***. Ne segue che si hanno **TROPPI FRAME** assegnati per quel processo, quindi *PIU' del dovuto*. Quindi alcuni di loro *possono* ***essere assegnati*** *ad altri processi*.
- Se si **sale sopra la soglia** *UB* (*superiore*) vuol dire che il numero di page fault è ***ALTO***. Ne segue che si hanno **POCHI FRAME** assegnati per quel processo, quindi *MENO* del dovuto. Quindi si devono ***aggiungere** nuovi frame da altri processi*

> ***REMINDER***
> Quando si entra in una località, si accede a qualcosa di nuovo, quindi si genereranno page fault obbligatori.
> Quando si è dentro la località, non dovrebbero esserci page fault perchè tutto è caricato in memoria
> Quando si esce dalla località, si generano nuovi page fault perchè si devono caricare nuovi dati dalla memoria



- All'ingresso della località si ha il numero massimo di page fault (PICCO MASSIMO nel grafico)
- Il picco indica quando ci si trova all'interno di una nuova località: vuol dire che tutto quello che serve è caricato in memoria.
- Si hanno **diversi picchi** perchè, essendo che non vengono caricati PEZZI DI PROGRAMMA, allora "le istruzioni successive" dello stesso programma devono essere caricate. Quindi avviene il ***PASSAGGIO FRA LOCALITA'***

Il **trashing** si riduce (*quasi a 0*) monitorando il numero di page fault che ogni processo esegue e mantenendosi all'interno dell'intervallo (*LB, UB*)

Politica di Pulitura

Per rendere i sistemi *più efficienti* bisogna garantire che ci sia un numero opportuno di ***frame liberi immediatamente disponibili*** in caso di page fault.

Vi è un **meccanismo** che spesso **svuota questi frame liberi** (*non è necessario farlo nell'immediato*, tipo quando si deve attendere una risposta e si è in attesa), se occupati: il meccanismo è eseguito da un **Processo Demone**, chiamato ***PAGING DAEMON***.

> Questo processo potrebbe,eventualmente, anche **ripescare tali frame in caso di richiesta** perchè, in teoria, quando si "svuota la memoria" si imposta un **FLAG** `libero` ma effettivamente non viene proprio svuotata. In questo caso è possibile recuperarne il contenuto cambiando il flag precedentemente impostato

Dimensione della pagina

- Solitamente questo parametro è scelto dal SO. In base alle dimensioni delle pagine ci sono vantaggi e svantaggi ma, comunque, sono sempre *potenze di 2*.
- La **dimensione della pagina** dipende dalla **tabela delle pagine**
- SI hanno bisogno di *più pagine* per memorizzare *più informazioni*
- La dimensione di una pagina è uguale alla dimensione di un frame, quindi più sono grandi i frame, più si riducono i page fault

Vantaggi di una pagina grande

- Si **minimizzano i page fault** visto che la dimensione della pagina è uguale alla dimensione del frame
- Richiede una **piccola tabella delle pagine**, cioè si devono registrare meno pagine
- Migliore **efficienza nel trasferimento I/O**.
- Maggiore è la **frammentazione interna**
- Si impiega più tempo per la fase di scrittura/lettura sulla pagina

Vantaggi di una pagina piccola

- Si ha una **minore frammentazione interna**
- Migliore risoluzione nel definire il working set in memoria perchè si ha *più dettaglio*
- Richiede la **tabella pagine grande**
- Si impiega meno tempo per la fase di scrittura/lettura sulla pagina

In generale...

Osservazione: in media **metà dell'ultima pagina viene sprecata**. Seguendo questa osservazione *conviene* usare una pagina di **piccole dimensioni**.

Esempio:

- Latenza=8ms
- Tempo di ricerca=20ms
- Tempo per trasferire 512b = 0,2ms
    - Se la pagina è 512, il trasferimento avviene in 28.2 ms
    - Se la pagina è 1024, il trasferimento avviene in 28.4