

## FRAME ALLOCATION > 10.5

È importante scegliere algoritmi che allochino il corretto # frame & processo.

Ogni processo necessita di un # minimo di frame associati per evitare rallentamenti eccessivi (Deciso da  $W_{\text{proc}}$ ).

### ALGORITMI DI ALLOCAZIONE 10.5.2

Ci sono 2 algo. principali:

FIXED ALLOCATION:  $n$  frame e  $S_{\text{proc}} = \frac{m}{n}$  frame  $\forall \text{proc}$

Divide i frame in parti uguali  $\forall \text{proc}$ .

- + vantaggi: - facile implem. - Il "resto" può essere frame pool
- svantaggi: Possibile assegn. eccessiva di frame.

### PRIORITY ALLOCATION

Divide i frame in maniera proporzionale alla sua size con la formula:

$$a_i = \frac{s_i}{S} \cdot m \quad \text{dove } a_i = \# \text{ frame al proc } i$$
$$S = \sum_{i=0}^n s_i \quad s_i = \text{Dim mem virt}$$

$m = \# \text{ di frame liberi}$

ESEMPIO Dati 62 frame su  $p_1$  e  $p_2$  dove richiedono

$$p_1 = 10, p_2 = 12 \text{ f}$$

[1] Calcolo i componenti della formula

$$m = 62 \quad S = 10 + 12 = 22 \quad S = 22$$

[2] Calcolo i tempi

$$a_1 = \frac{s_1}{S} \cdot m = \frac{10}{22} \cdot 62 \approx 4$$

$$a_2 = \frac{s_2}{S} \cdot m = \frac{12}{22} \cdot 62 \approx 5 \text{ f}$$

- + vantaggi: Ogni proc ha un # frame fisso

- svantaggi: Non tiene conto delle priorità dei proc  $\Rightarrow$  mod. formula

## EFFETTI DEL PAGE REPLACEMENT > 10.5.3

Gli effetti del Page Replacement influiscono sul Frame Alloc.

si definiscono 2 strategie:

### GLOBAL REPLACEMENT

Nella Page Repl. il processo può scegliere fra tutti i frame, inclusi quelli di altri processi

- + vantaggi: - facile integr. con Priority Sched.
  - # Pagine per il processo può crescere indirett.
- svantaggi: - Durata dipende dagli altri processi

### LOCAL REPLACEMENT

Il Processo può scegliere solo fra le sue Pages:

- + vantaggi: La durata del proc. dipende solo da se
- svantaggi: Possibile preclusione page meno usate.

### REAPER ROUTINE ➤

Per mantenere il min # frames, l'S.O. avvia una "reaper routine" per fare Page Repl.

se il # di free frames < soglia. Approx LRU

### NUMA ➤ 10.5.4

Nei sistemi NUMA, gli algoritmi di Frame Alloc. e Page Repl. devono minimizzare il # Hop fra le memorie.

Bisognerebbe anche ri-schedulare i processi sulla stessa CPU, facilitando le cache hit per gli accessi.

### TRASHING ➤ 10.6

Aumentamento che si verifica quando un processo spende più tempo nel Paging che nella sua esecuzione.

Esempio:  $P_1$  ha finito i frame e deve fare Page Repl. delle sue pages ma gli servono tutte.

### LEGARLE CON MULTIPROG.

Aumentando troppo la multiprog., si incorre in uno spiraglio di Trashing, risolvibile dimin.

Esempio:  $P_1$  ha  $F$  frame, ne chiede  $F_1 > F$  ma free frames=0 quindi SO effettua Page Repl. → Vittima si mette in wait queue → SO pensa ci siano podi proc. e v

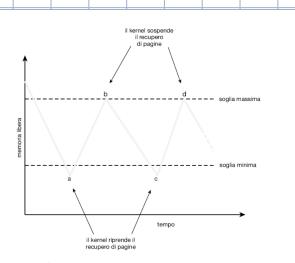


Figura 10.18 Recupero di pagine.

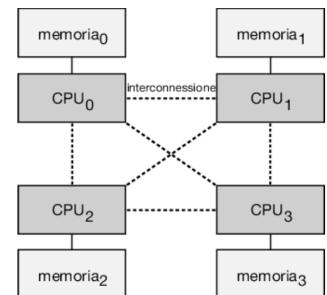


Figura 10.19 Architettura multiprocesso numa.

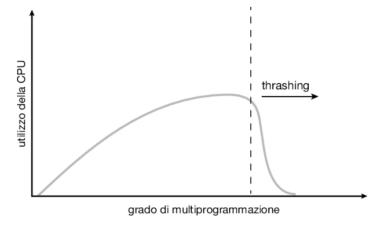


Figura 10.20 Thrashing.

Possibili soluzioni al problema possono essere:

- Local Paging:

Isola il Thrashing, ma i processi causano attese maggiori nella coda di pag.

- Priority Frame allocation

Soluzioni al Thrashing necessitano del concetto di

LOCALITÀ → Necessario per working set

Rappresenta i frame in uso per un determinato proc.

L'uso della memoria di un processo può essere diviso in località distinte che variano nel tempo.

Esempio in T(a):  $L = \{ \underline{18}, \underline{19}, \underline{20}, 21, 22, 23, 24, 24, 30 \}$

= T(b):  $L = \{ \underline{18}, \underline{19}, \underline{20}, 24, 25, 26, 27, 28, 29, 31, 32, 33 \}$

Se non si riesce a soddisfare la località → Possibile Thrashing.

### WORKING SET 10.6.2

Consente di approssimare la località esaminando i frame usati dal processo. Si lavora con  $\Delta = \#$  frame da guardare

#### ESEMPIO

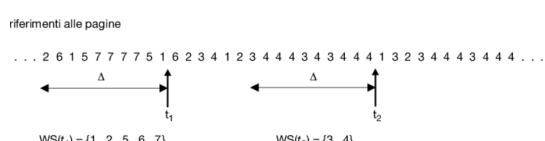


Figura 10.22 Modello del working set.

La scelta di  $\Delta$  è cruciale:

- $\Delta$  troppo piccolo : Non include l'intera località.
- $\Delta$  troppo grande : Le località si sovrappongono.

### DIMENSIONE E THRASHING (RISUARDO)

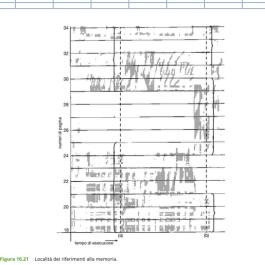
Si può determinare la dim. totale dei # frame richiesti /proc:

$$D = \sum Wset_i \quad \forall i \in Proc$$

Se  $D > m$  dove  $m = \#$  frame disp. → Thrashing

### FUNZIONAMENTO CON SO

1. L'SO calcola il Wset / proc



2.? Se # frame > min

? T Aumenta multiprog.

? F Sposta un proc. in mem. secondaria → Page liberate

La difficoltà sta nel tenere conto che i WS possono variare a ogni riferimento. Per fare questo si approssima con timer

### PAGE FAULT FREQUENCY 10.6.3

Metodo alternativo al working set, che limita

il trashing analizzando la frequenza di page fault.

- troppi P.F. = Il processo necessita frame
- pochi P.F. = Il processo ha troppi frame

Definendo limiti superiori ed inferiori si riesce

a bilanciare. Se non ci sono free frames → Swap mem sec.

### PF. FREQ SU WORKING SET

La P.F. freq è interessata da:

- Picchi: Quando si richiedono le page per la località.
- valli: Quando le page per la località sono sufficienti.

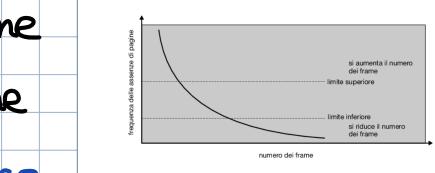
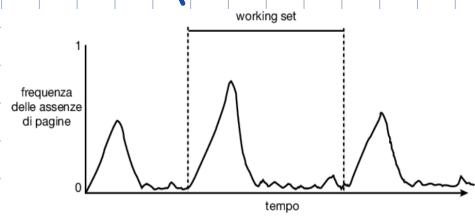
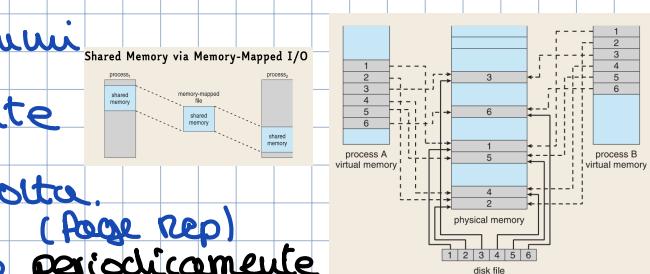


Figura 10.23 Frequenza dei page fault.



### MEMORY MAPPED FILES >

Gli accessi all'I/O vengono efficientati riservando aree di memoria primaria in cui i programmi si collegano (mmap). Questo consente di evitare passaggi nel disco ogni volta.



Le scritture avvengono alla fine o periodicamente (page rep)

### MEMORIA DI MASSA > 11

La memoria di massa descrive la memoria secondaria:

### DISCHI MAGNETICI / HARD DISKS

Sono formati da un insieme di dischi a cui è appoggiata una testina di r/w.

Metriche utili:

- quanto dura • RPM • Transfer rate

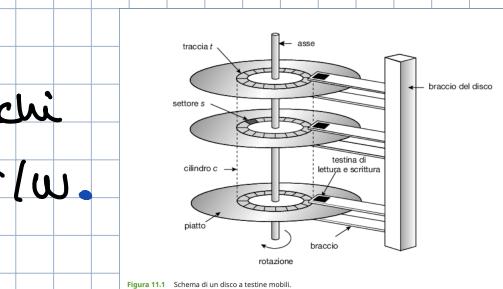


Figura 11.1 Schema di un disco a testina mobile.

- Seek Time: Il tempo per spostare la Testina nel settore
- Access Latency: Latenza di accesso
- Banda di Trasf: Quanta info. si può trasferire in tempo.

Ogni disco è formato da settori che tagliano a fette i dischi.

Hard Disk Performance
Access Latency = Average access time = average seek time + average latency
• For fastest disk 3ms + 2ms = 5ms
• For slowest disk 5ms + 5,50ms = 14.50ms
• Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead
• For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 400MB/s transfer rate with a 1ms controller overhead =
• 5ms + 4.3ms + 0.1ms + transfer time =
• Transfer time = $4KB / 100MB/s * 800 / 60 * 1024KB = 32 / (1024) = 0.031ms$
• Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

## SSD/HDD

Sono dispositivi elettrici di memorizzazione usati spesso in mobile

- + pros: no parti meccaniche → più durabili fisicamente
- cons: durabilità della memoria ridotta

## PERFORMANCE

Le SSD possono essere

- NAND Flash: + Banda, + Latenza (più sviluppata)
- NOR Flash: - Banda, - Latenza

## MAGNETIC TAPE / NASTRI

Usati in contesti specifici, principalmente per dati a medio term lungo

- + pros: A temp. giusta, consentono memorizzazione a lungo term
- cons: Accessi a memoria impiegano minuti/ore (meccanico)

## DISK ATTACHMENT > 11.7

I dischi si collegano sul Bus con diversi protocolli:

- SCSI : Bus che consente la comunicazione fino a 16 elementi. ( SCSI Initiator fa le req, target tasks)
- FC : Architettura seriale ad alta velocità (Fibre Cable)

I dispositivi collegati si distinguono coi Logical Unit Disks

## STORAGE ARRAY / RAID (Redundant Array of Inexpensive V)

Uso di più dischi insieme per maggiore performance o per sicurezza in maniera contemporanea.

Per accedervi, ci si collega a uno

- SAN (Storage Area Network):

Esporta al driver del SO uno storage (HW)

## NAS (Network Attached Storage)

Espone al driver del SO come un elem del F.S.

SCHEDULING DEI DISCHI > 11.2

Si intende la policy atta a definire lo scheduling delle letture degli elementi sul disco in modo da:

- min. Latency: minimizzare la latenza. (spostam. testina)
- max. Bandwidth: la Velocità di accesso sul disco.

Ogni richiesta necessita di:

- modalità di accesso
- Indirizzi del disco e mem. primaria
- # settori interessati

Le richieste vengono organizzate in una coda di richieste, definita  $\forall$  dispositivo o  $\forall$  disco  $\forall$  dispositivo:

### ALG: FCFS

Prevede di seguire l'ordine di arrivo:

- soggetto ai valori  $\rightarrow$  male performance
- + facile da implementare

### SHORTEST SEEK TIME FIRST

Muovere la testina verso il più vicino

- + meno latenza
- starvation su richieste lontane

### SCAN o dell'"Ascensore"

Prevede di andare verso un estremo del disco, per poi ripartire verso l'altra estremità.

- + latenza dim. nel primo giro
- finito giro, latenza aumenta

### CSCAN

Variante di SCAN che prevede di fare un salto unico verso la 2<sup>a</sup> estremità e ripartire.

- + Bande

- + Latenza, soprattutto quando si cambia.

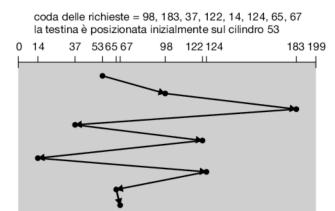
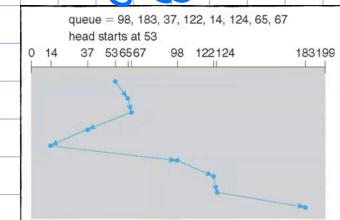


Figura 11.6 Scheduling fcfs.

### Testina spostata di GKO



= 236

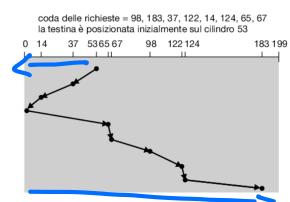


Figura 11.7 Scheduling scan.

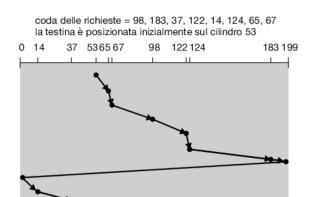


Figura 11.8 Scheduling cscan.