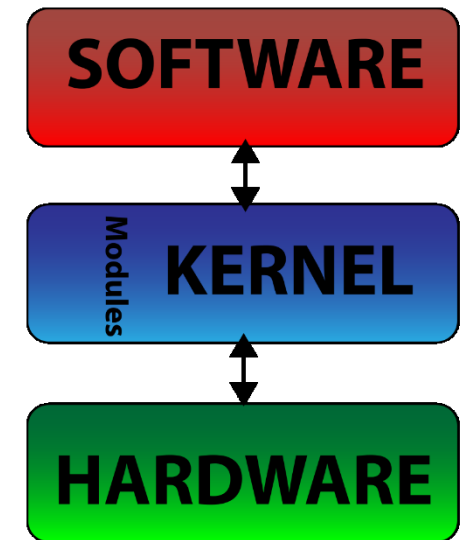
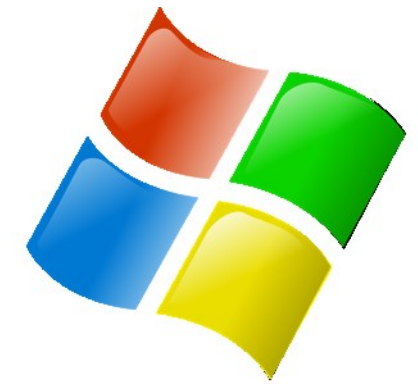




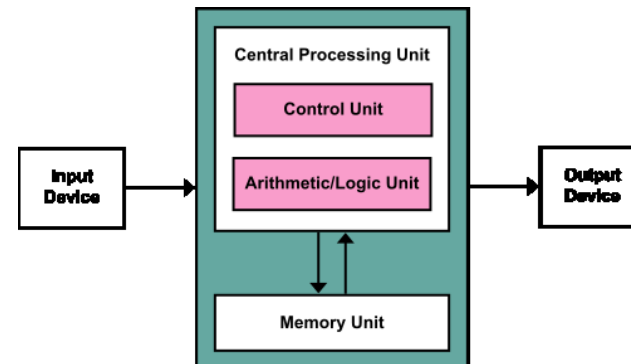
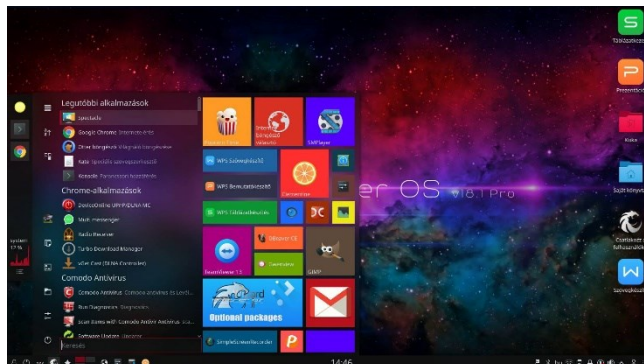
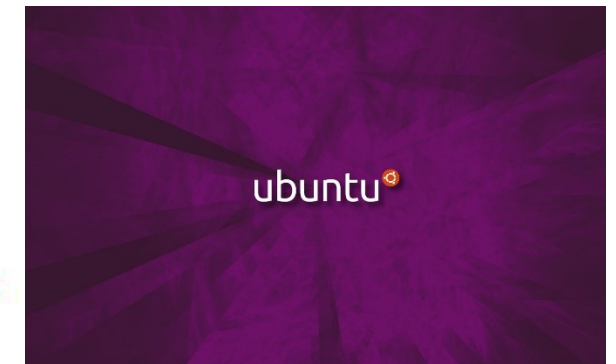
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

*Corso di Sistemi Operativi
A.A. 2019/20*

Allocazione di memoria



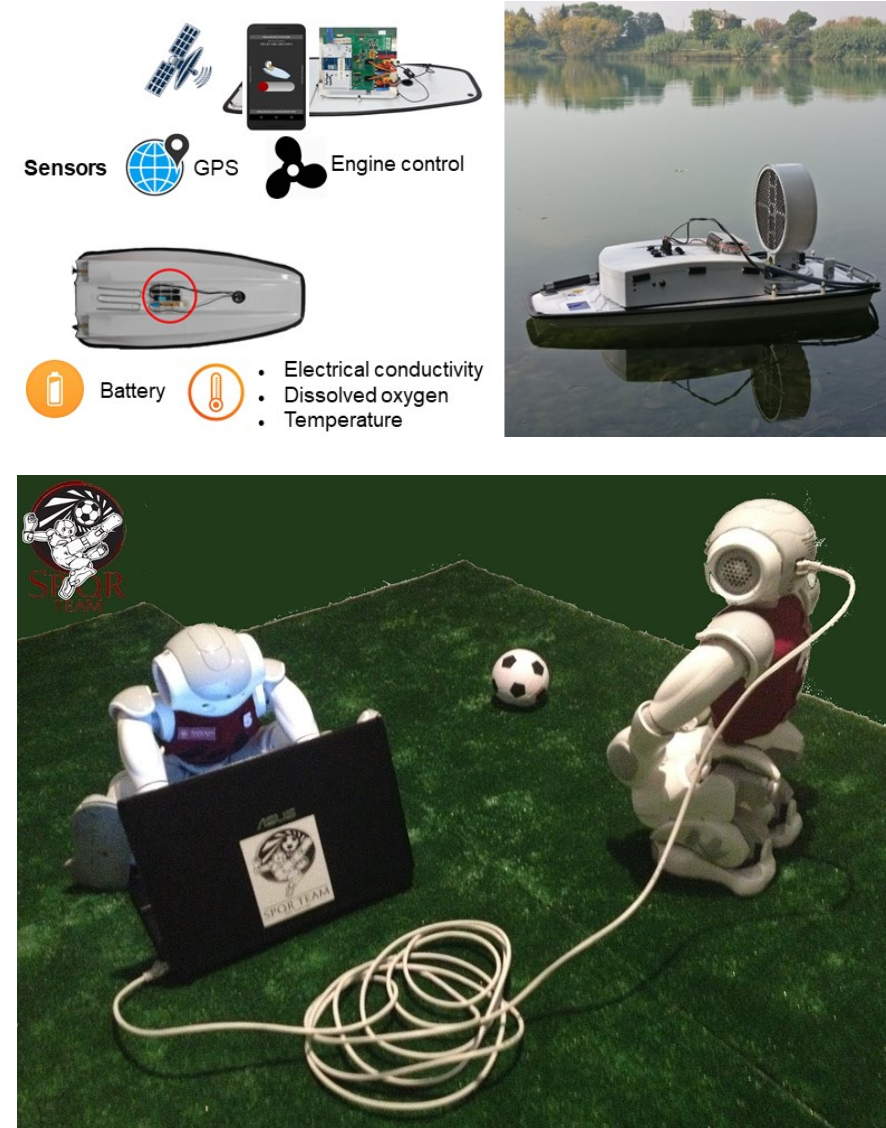
Docente:
**Domenico Daniele
Bloisi**



Novembre 2019

Domenico Daniele Bloisi

- Ricercatore RTD B
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Ricevimento

- In aula, subito dopo le lezioni
- Martedì dalle 11:00 alle 13:00 presso:
Campus di Macchia Romana
[Edificio 3D](#) (Dipartimento di Matematica,
Informatica ed Economia)
[Il piano, stanza 15](#)

Email: domenico.bloisi@unibas.it



Programma – Sistemi Operativi

- Introduzione ai sistemi operativi
- Gestione dei processi
- Sincronizzazione dei processi
- Gestione della memoria centrale
- Gestione della memoria di massa
- File system
- Sicurezza e protezione

Allocazione dei frame

Numero minimo di frame

Vincoli dell'allocazione dei frame:

1. Non si possono assegnare più frame di quanti siano disponibili
2. È necessario assegnare almeno un **numero minimo di frame**



prestazioni

Algoritmi di allocazione dei frame

allocazione
uniforme

allocazione
proporzionale

allocazione
globale

allocazione
locale

Recupero

Strategia per implementare una **politica globale** di sostituzione delle pagine:

garantire che ci sia sempre sufficiente memoria libera per soddisfare nuove richieste

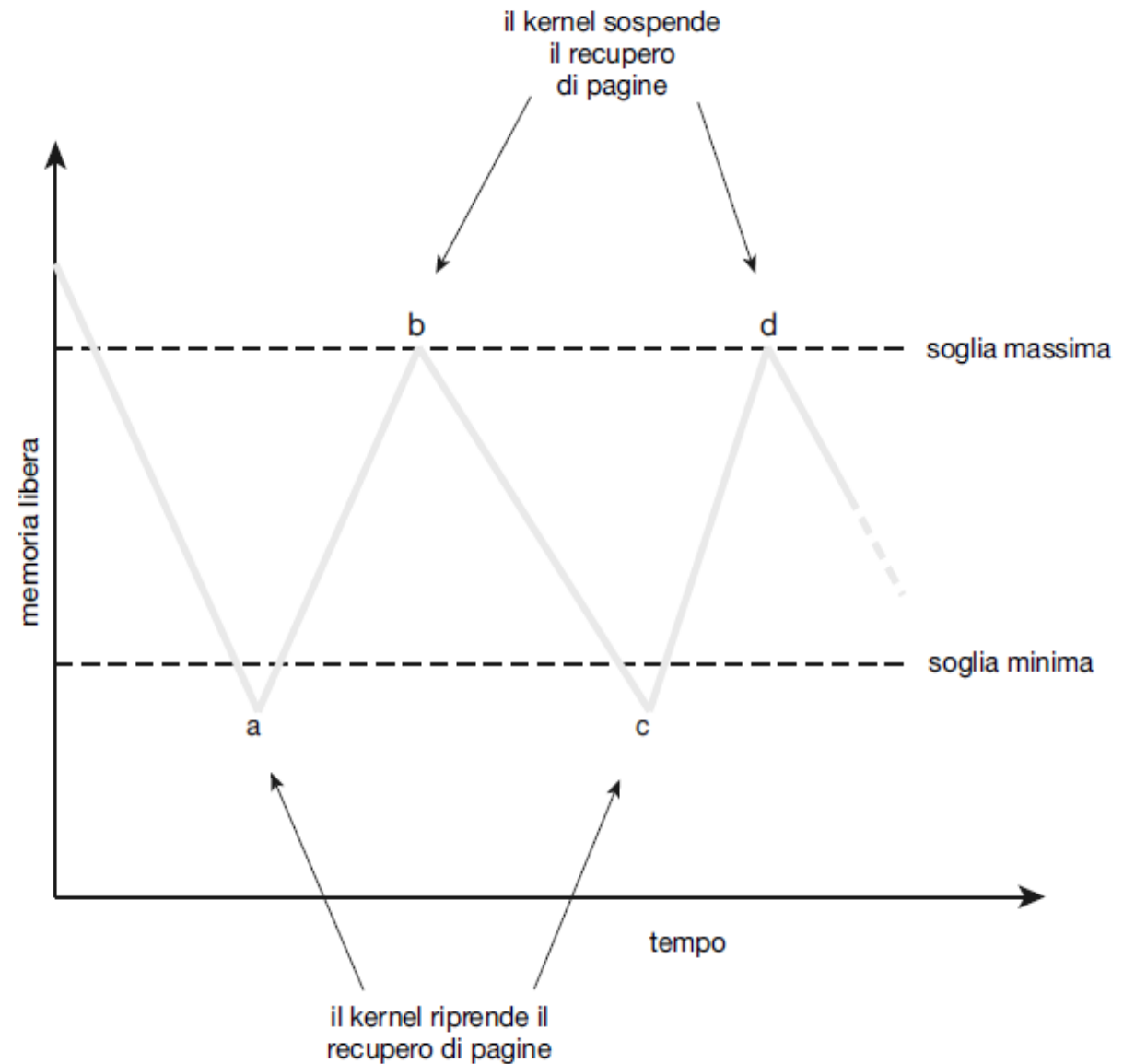


Figura 10.18 Recupero di pagine.

Sistemi con accesso non uniforme in memoria NUMA

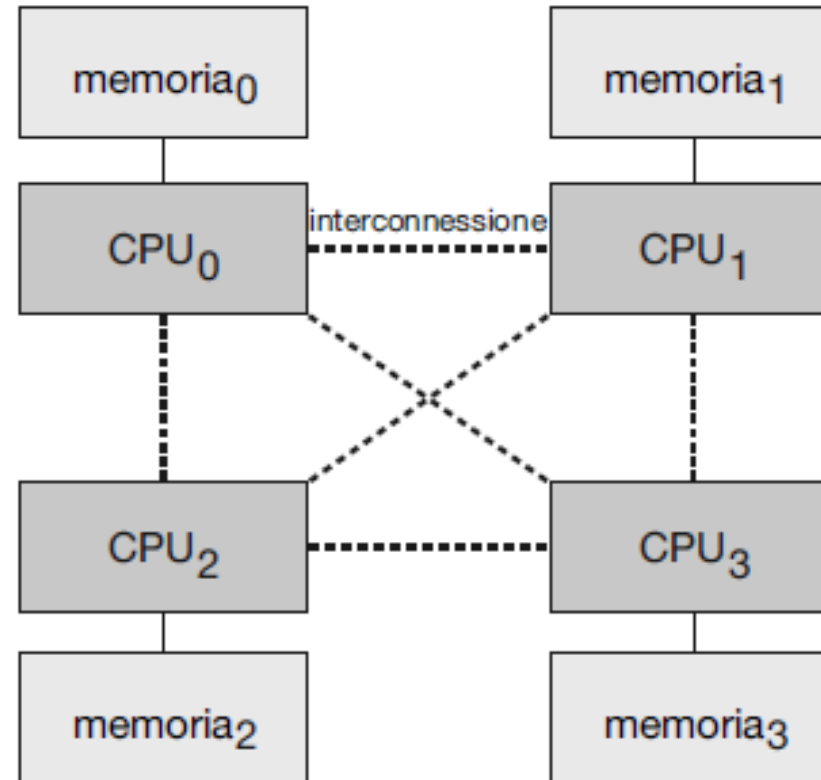


Figura 10.19 Architettura multiprocesso NUMA.

Trashing

Il **thrashing** si verifica quando un sistema spende più tempo per la paginazione rispetto al tempo destinato all'esecuzione.

Il **thrashing** causa notevoli problemi di prestazioni.

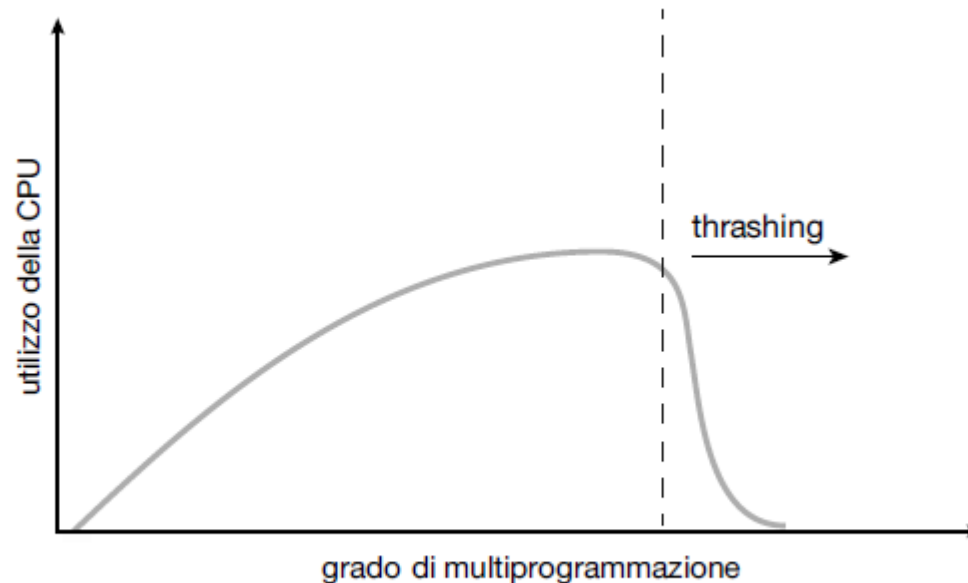


Figura 10.20 Thrashing.

Località

Una **località** è un insieme di pagine usate attivamente insieme

La Figura 10.21 illustra il **concetto di località** e come la località di un processo cambia nel tempo

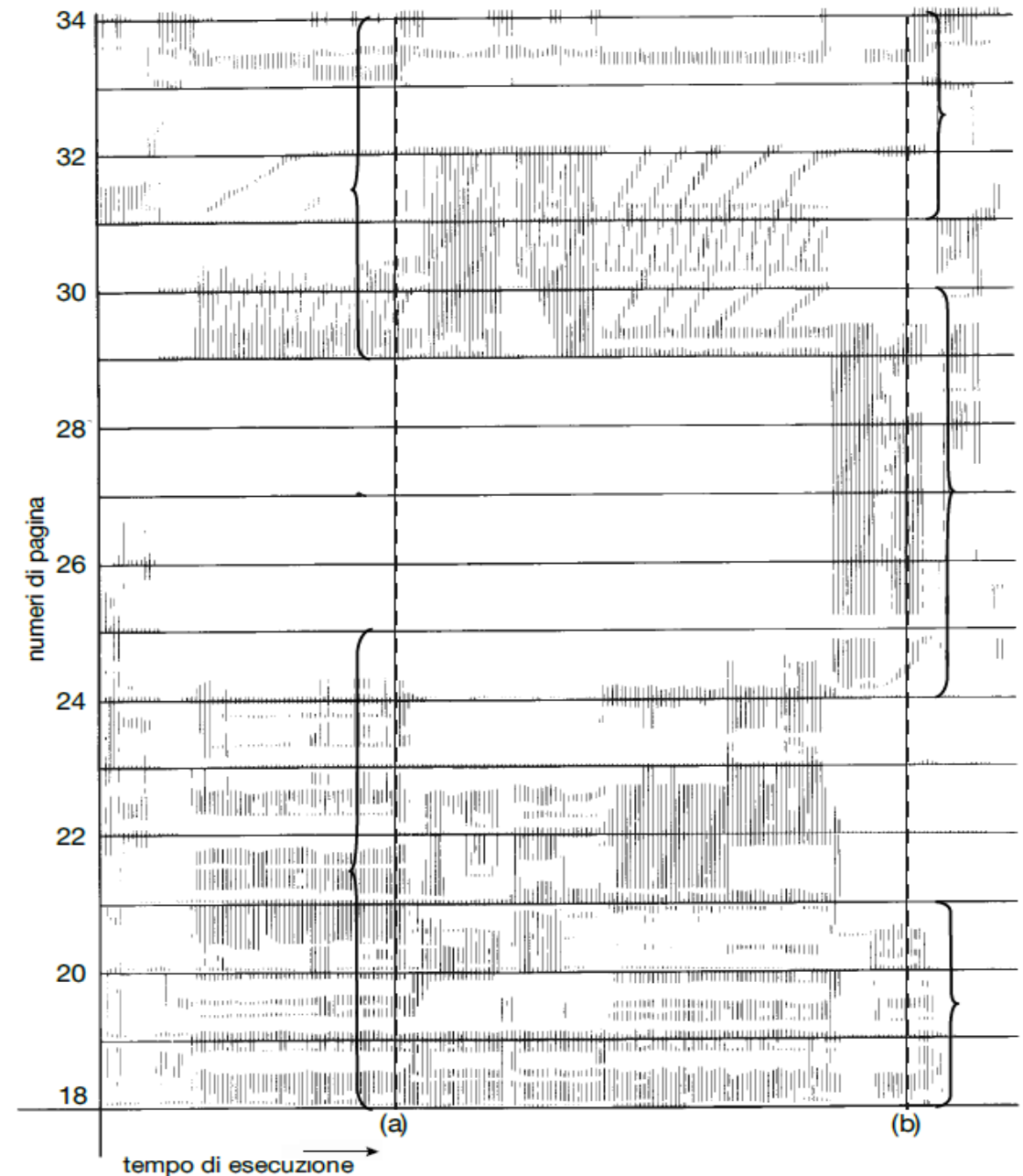


Figura 10.21 Località dei riferimenti alla memoria.

Modello del working set

L'insieme di pagine nei più recenti Δ riferimenti è il **working set** → *l'insieme di pagine utilizzate da un processo in un dato istante.*

Se una pagina è in uso attivo si trova nel **working set**; se non è più usata esce dal **working set** Δ unità di tempo dopo il suo ultimo riferimento. Quindi, il **working set** non è altro che un'approssimazione della **località** del programma.

riferimenti alle pagine

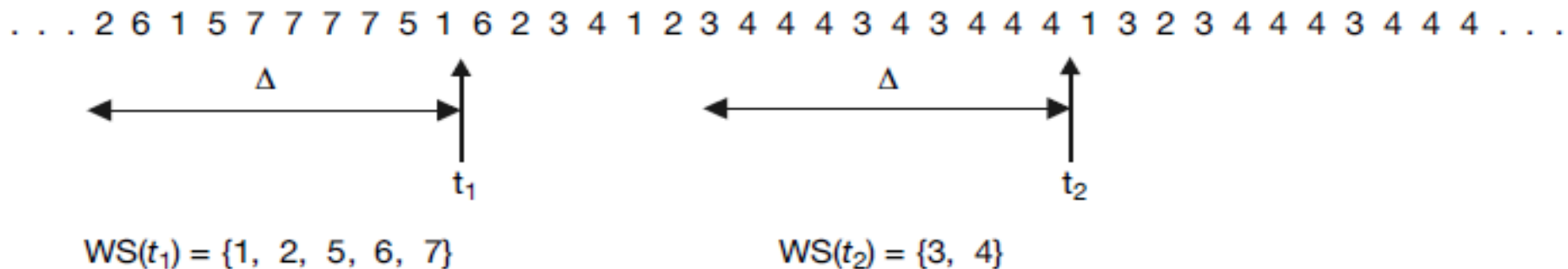


Figura 10.22 Modello del working set.

Frequenza dei page fault

Si può fissare un *limite inferiore* e un *limite superiore* per la **frequenza** desiderata dei **page fault**. Se la frequenza effettiva dei page fault per un processo oltrepassa il limite superiore, occorre allocare a quel processo un altro frame; se la frequenza scende sotto il limite inferiore, si sottrae un frame a quel processo → prevenire il **thrashing**

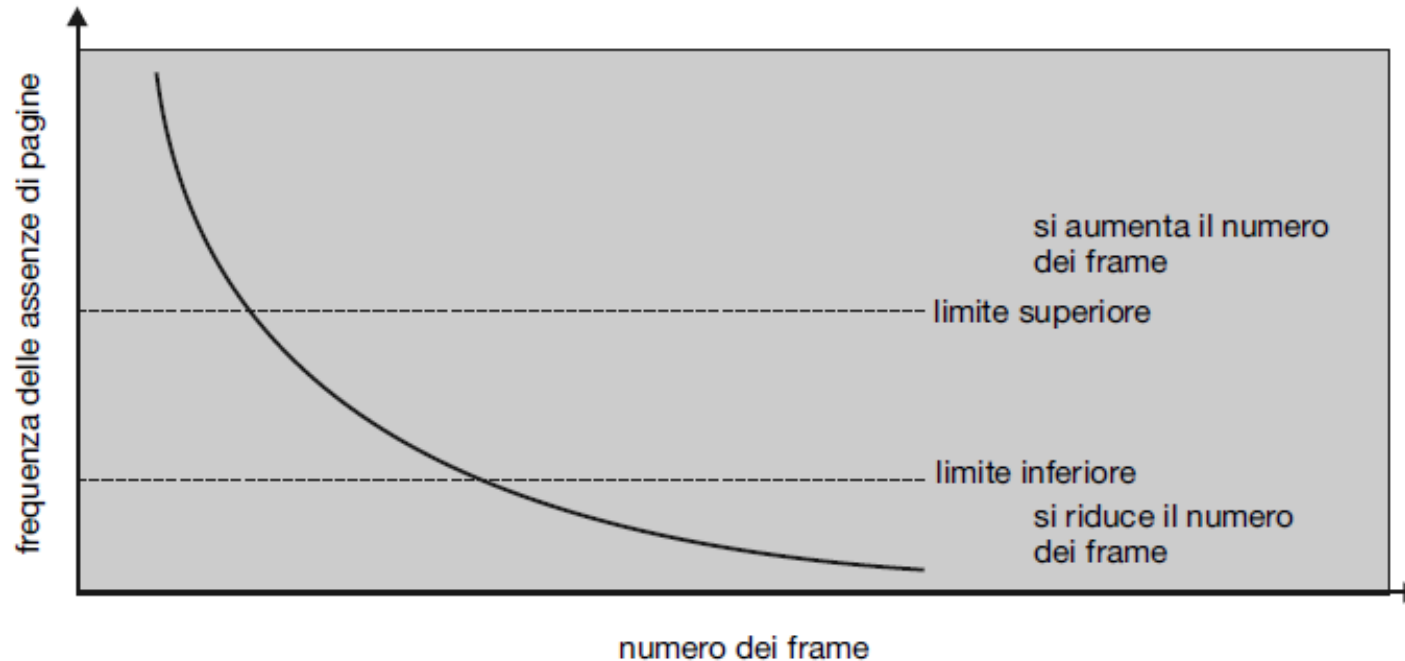


Figura 10.23 Frequenza dei page fault.

Compressione della memoria

La **compressione della memoria** è una tecnica di gestione della memoria che consiste nel comprimere un certo numero di pagine in una singola pagina

La **memoria compressa** è un'alternativa alla **paginazione** e viene utilizzata su sistemi mobili che non supportano la paginazione

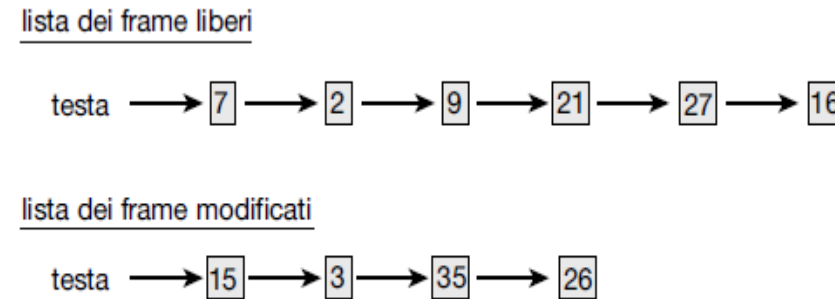


Figura 10.24 Lista dei frame liberi prima della compressione.

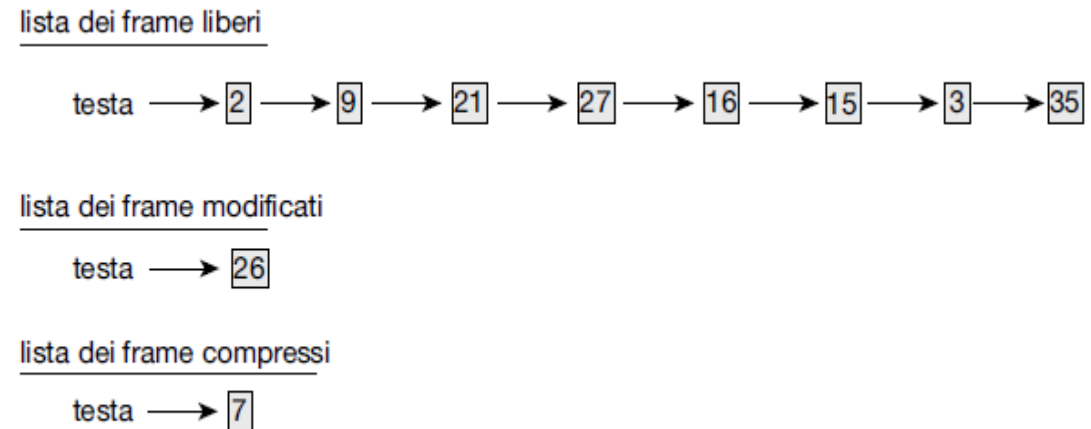



Figura 10.25 Lista dei frame liberi dopo la compressione.

Allocazione di memoria del kernel

La **memoria del kernel** è allocata in modo differente rispetto a quanto avviene per i processi in modalità utente, utilizzando blocchi contigui di dimensioni variabili.

Due tecniche comuni per l'**allocazione della memoria del kernel** sono:



Sistema
buddy

Allocazione
a lastre
(slab)

Allocazione di memoria del kernel

Sistema buddy

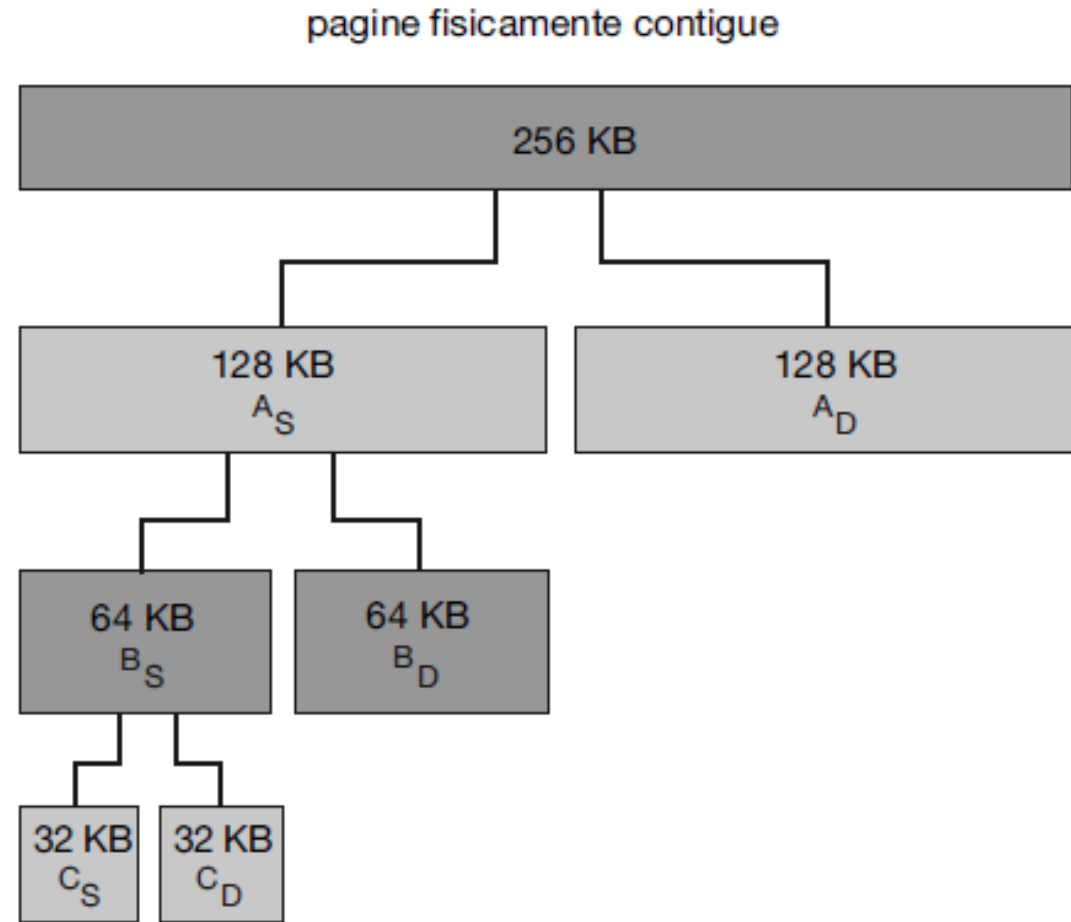


Figura 10.26 Sistema di allocazione buddy.

Allocazione buddy

- Buddy Allocator viene usato per allocare oggetti di dimensione variabile.
- Il buffer viene partizionato ricorsivamente in 2, creando di fatto un albero binario.
- La foglia più piccola che soddisfa la richiesta di memoria sarà ritornata al processo.
- Il buddy associato a una foglia sarà l'altra regione ottenuta dalla divisione del parent.
- Se un oggetto è più piccolo della minima foglia che lo contiene, il restante spazio verrà sprecato.
- Quando un blocco viene rilasciato, esso verrà ricompattato con il suo buddy (se libero), risalendo fino al livello più grande non occupato.

Allocazione di memoria del kernel

Allocazione a lastre (SLAB)

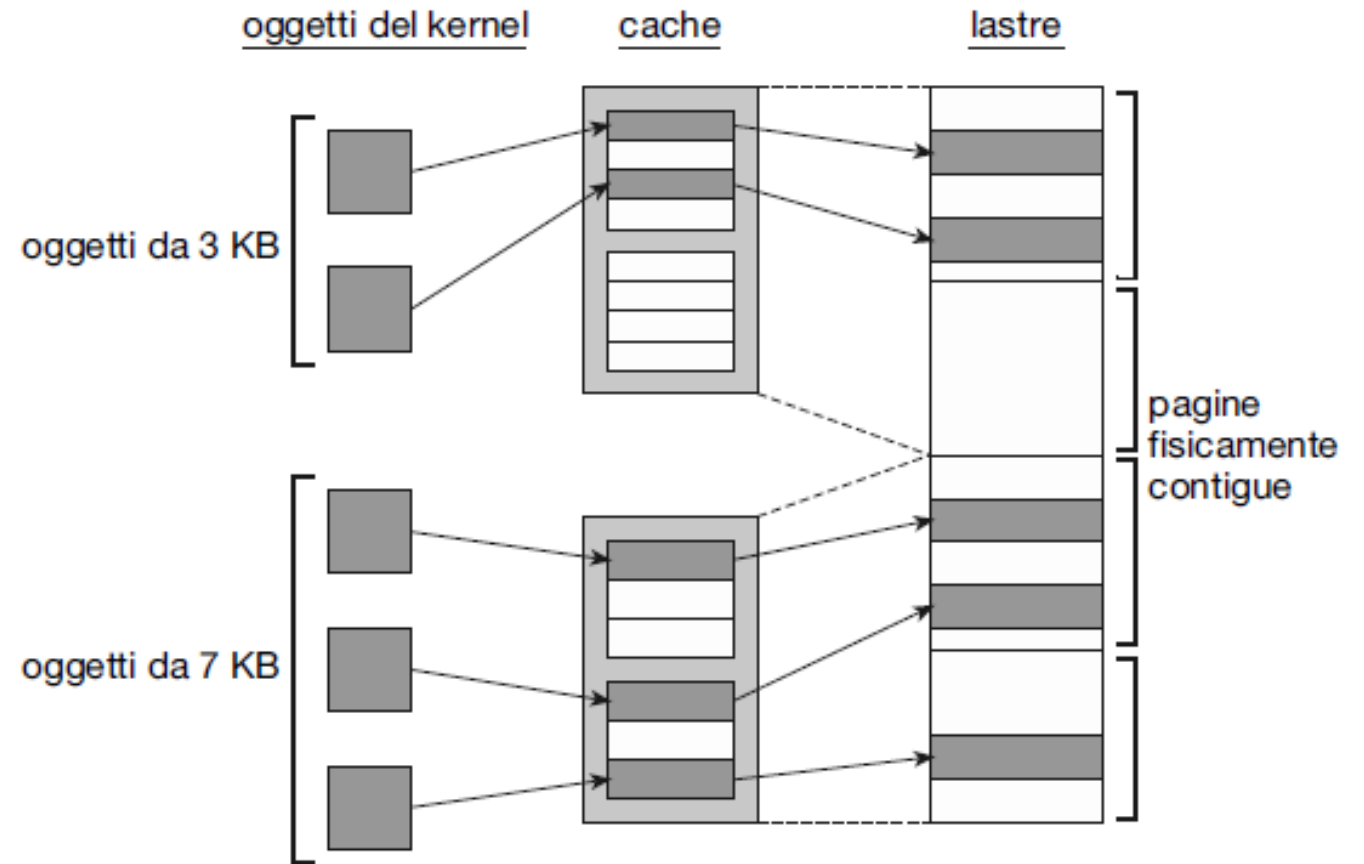


Figura 10.27 Allocazione a lastre (slab).

Allocazione SLAB

- Slab Allocator viene usato per allocare oggetti di dimensione fissa.
- Può allocarne fino ad un numero massimo fissato.
- Il buffer viene diviso in chunk di dimensioni item size.
- Per poter organizzare il buffer viene usata una struttura ausiliaria che tenga l'indice dei blocchi ancora liberi.
- Una array list soddisfa tale richiesta.

Portata del TLB

Tasso di successi (*hit ratio*) di un TLB → percentuale di traduzioni di indirizzi virtuali risolte dal TLB anziché dalla tabella delle pagine → proporzionale al numero di elementi del TLB

Portata del TLB → numero di elementi moltiplicato per la dimensione delle pagine

La **portata del TLB** esprime la quantità di memoria accessibile dal **TLB**

Una tecnica per aumentare la portata del **TLB** è aumentare la dimensione delle pagine.

Esempi di sistemi operativi

Realizzazione della memoria virtuale in:



Linux

Windows

Solaris

Linux, Windows e Solaris gestiscono la memoria virtuale in modo simile, utilizzando, tra l'altro, la **paginazione su richiesta** e la **copia su scrittura**.

Ogni sistema utilizza anche una **variante per approssimazione di LRU** nota come **algoritmo a orologio**.

Linux

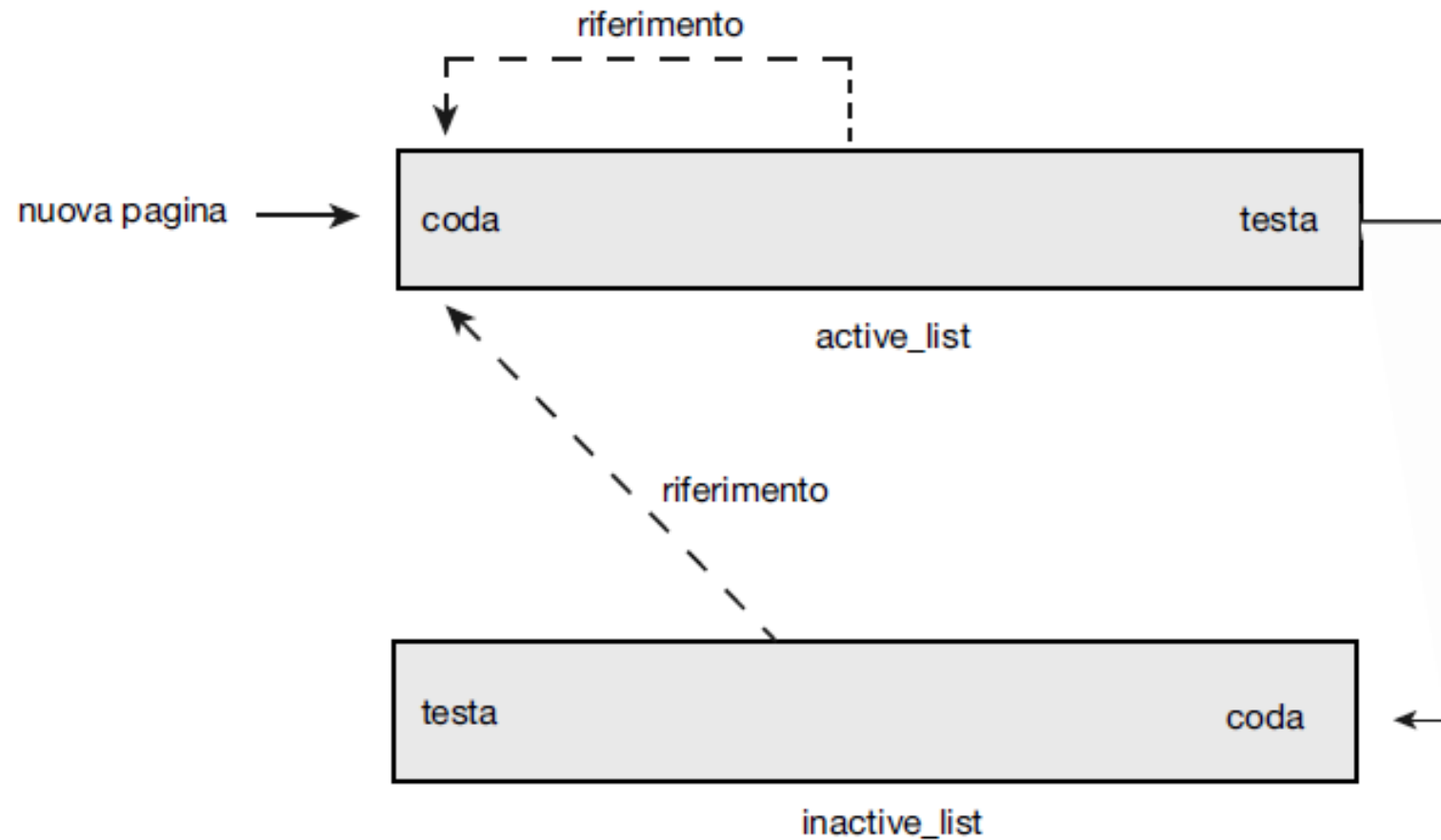


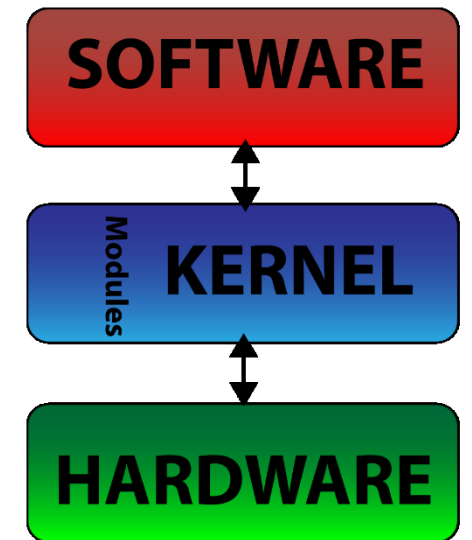
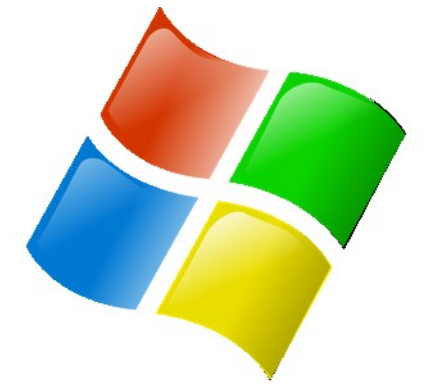
Figura 10.29 Le strutture `active_list` e `inactive_list` di Linux.



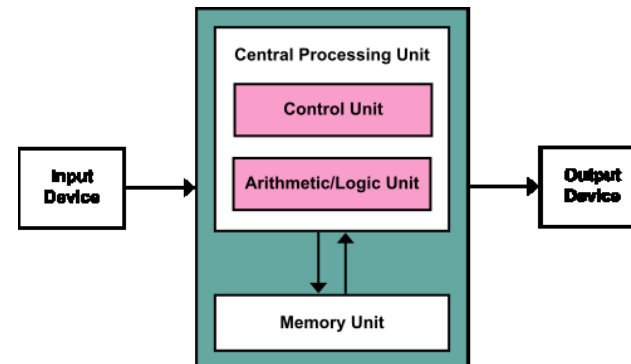
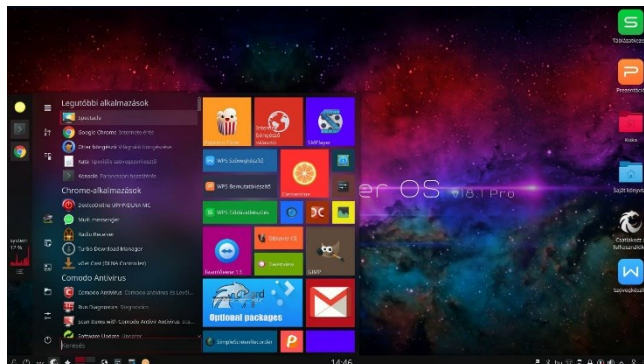
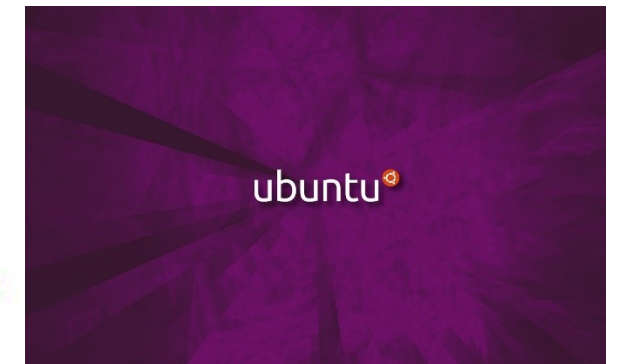
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

*Corso di Sistemi Operativi
A.A. 2019/20*

Allocazione di memoria



Docente:
**Domenico Daniele
Bloisi**



Novembre 2019