

Sistemi Operativi

C.d.L. in Informatica (laurea triennale)
Anno Accademico 2023-2024

Canale A-L

Dipartimento di Matematica e Informatica – Catania

File System e Dischi

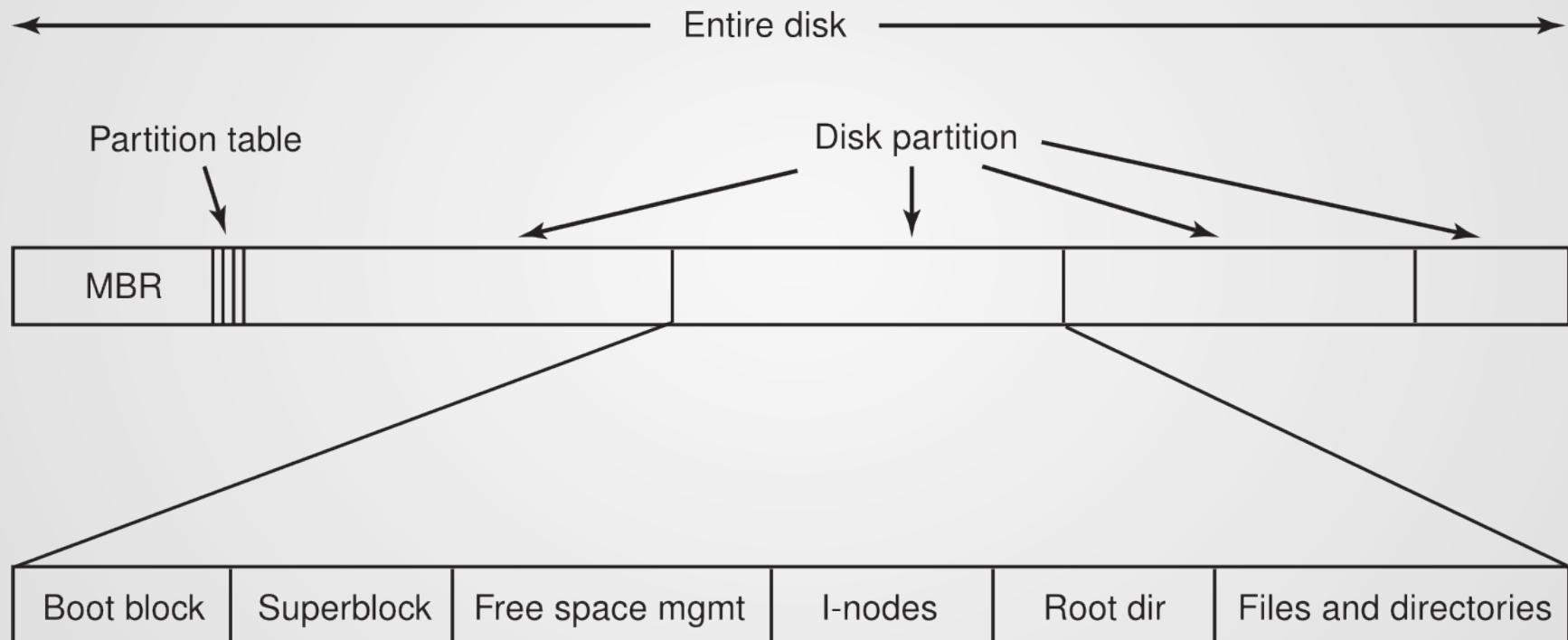
Prof. Mario Di Raimondo

I file system

- Problema di base: gestire **grandi quantità** di informazioni, in modo **persistente** e condiviso **tra più processi**;
- astrazione: **file** e **directory**;
- i dettagli di gestione ed implementazione costituiscono il **file system**;
- esempi di **dettagli**:
 - nomenclatura;
 - tipi di file;
 - tipi di accesso;
 - metadati (o attributi);
 - operazioni supportate sui file;
 - accesso condiviso ai file: i **lock**:
 - shared vs. exclusive;
 - mandatory vs. advisory;
- **strutture dati** per la gestione dei file: globale e per processo.

Struttura di un file system

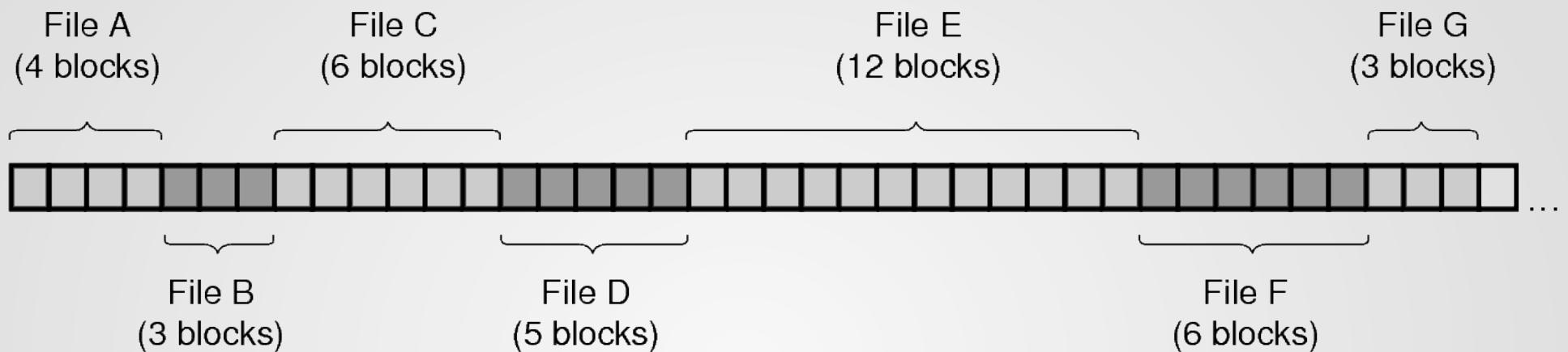
- **Master Boot Record (MBR);**
- partizioni e **boot record** (o boot block);
- **superblocco;**



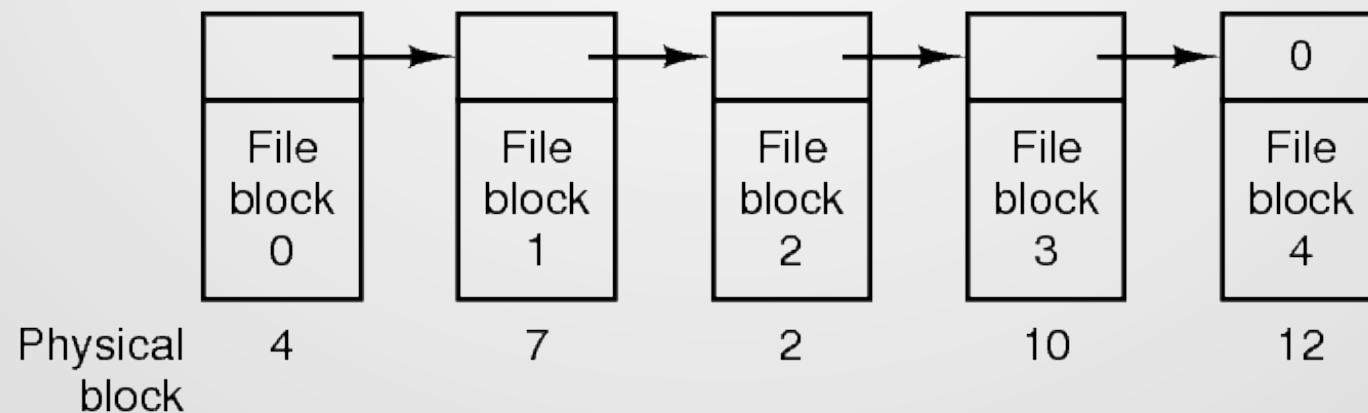
- layout moderno alternativo: **GPT** (GUID Partition Table) definito dallo standard **EFI**.

Implementazione dei file

- **Allocazione contigua.**

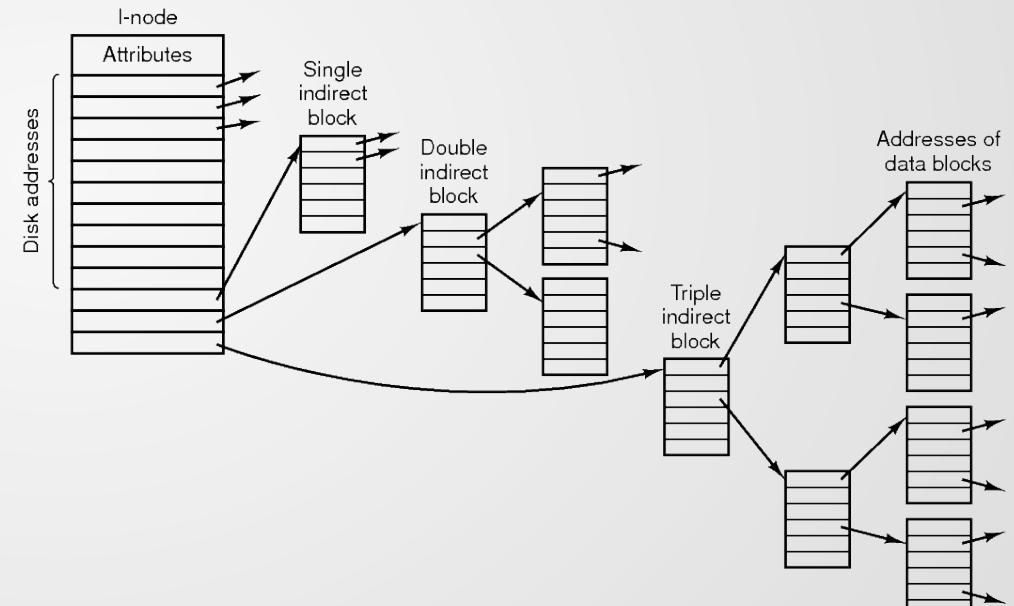
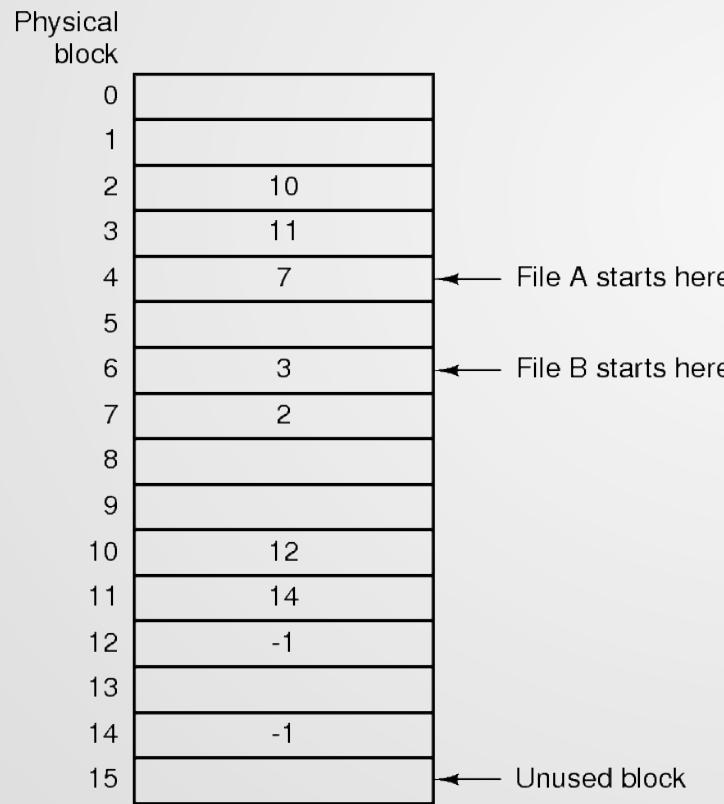


- **Allocazione con liste collegate (o allocazione concatenata).**



Implementazione dei file

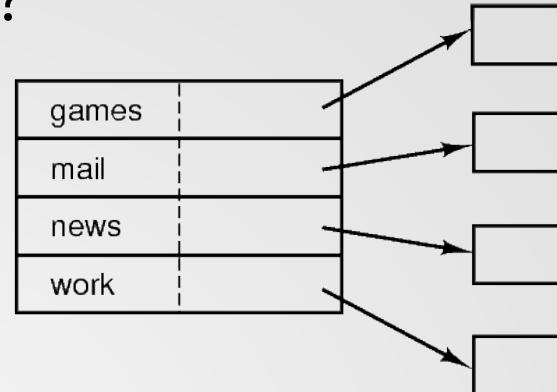
- **Allocazione con liste collegate su una tabella di allocazione dei file (file allocation table – FAT) (o allocazione tabellare).**
- **Allocazione con nodi indice (index node – i-node) (o allocazione indicizzata):**
 - varianti: **collegata, multilivello, ibrida.**



Implementazione delle directory

- Dove memorizzare i **metadati/attributi?**

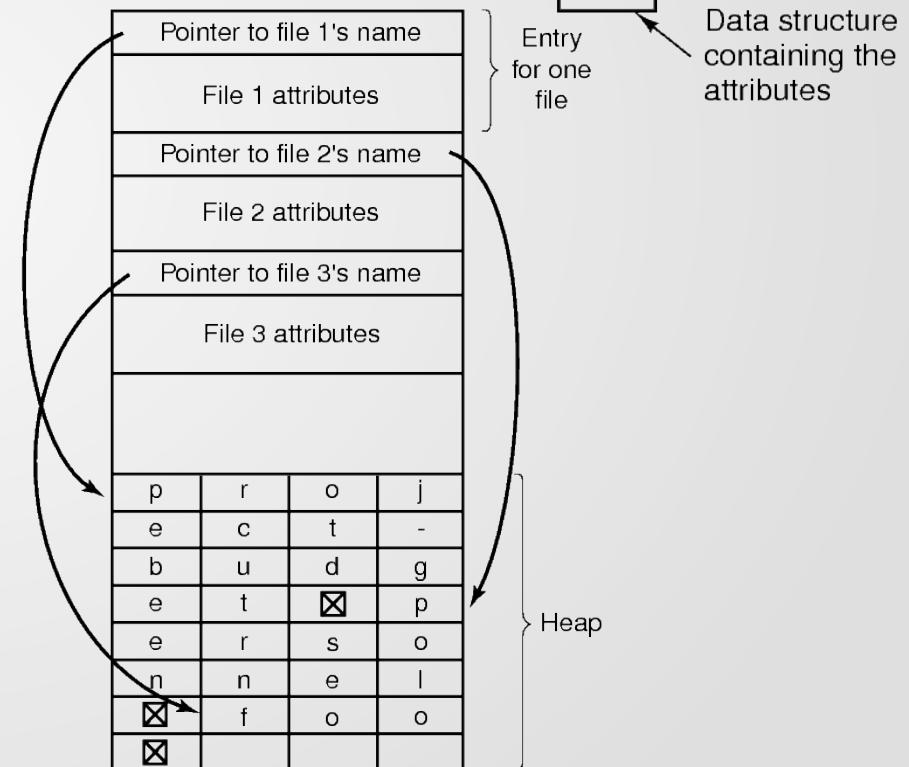
games	attributes
mail	attributes
news	attributes
work	attributes



- nomi lunghi:**
 - lung. variabile;
 - tramite heap;
- prestazioni:**
 - tabella hash;
 - cache.

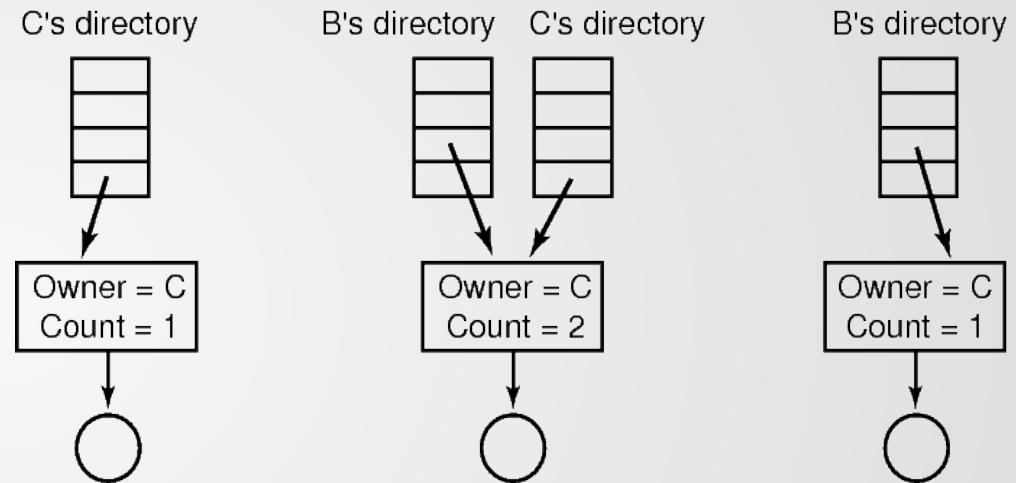
Entry for one file

File 1 entry length			
File 1 attributes			
p	r	o	j
e	c	t	-
b	u	d	g
e	t	☒	
File 2 entry length			
File 2 attributes			
p	e	r	s
o	n	n	e
l	☒		
File 3 entry length			
File 3 attributes			
f	o	o	☒
⋮			



Condivisione di file su un file system

- **Scenario:** due o più utenti vogliono condividere un file;
- usando una FAT: duplicare la lista con i riferimenti ai blocchi;
 - problemi in caso di append;
- usando i-node: **hard-link**;
 - contatore dei link;
 - anomalia con accounting;
- ulteriore approccio: **soft-link**;
 - universale e permettere di fare riferimenti al di fuori del file system;
 - appesantimento nella gestione;
- eventuali **problemi** in fase di attraversamenti e backup.



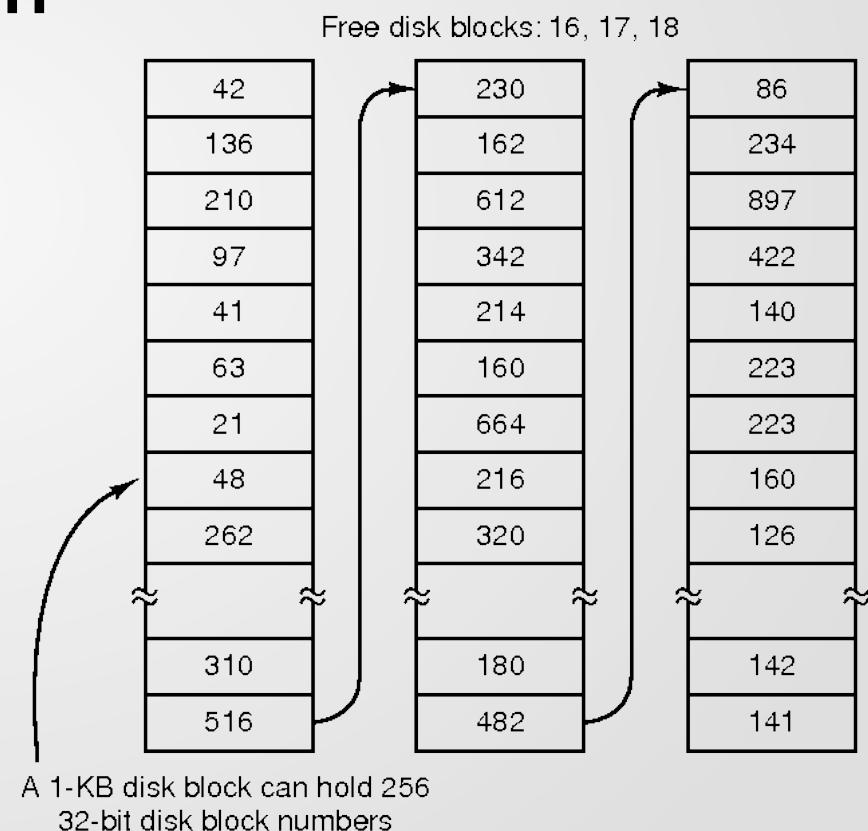
Gestione blocchi liberi

- Attraverso l'uso di una **bitmap**:
 - relativamente **piccola**;
 - strategie di **allocazione in memoria**:
 - tutta in memoria, o;
 - un blocco alla volta;
 - paginata con VM;

1001101101101100
0110110111110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
~ ~
0111011101110111
1101111101110111

Gestione blocchi liberi

- attraverso l'uso di **liste concatenate**:
 - richiede più spazio;
 - ma si sfruttano i blocchi stessi liberi;
 - possibilità di inserire **contatori** per blocchi contigui.



Controlli di consistenza

- A seguito di **crash** del sistema i file-system possono diventare inconsistenti;
- apposite **utility** possono effettuare dei **controlli di consistenza**:
 - sui **blocchi**;

Block number	Blocks in use
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 0 0 1 1 1 1 0 0

0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1	Free blocks
---------------------------------	-------------

(a)

Block number	Blocks in use
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0

0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 1	Free blocks
-----------------------------------	-------------

(b)

Block number	Blocks in use
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 0 0 1 1 1 1 0 0

0 0 1 0 2 0 0 0 0 1 1 0 0 0 1 1	Free blocks
---------------------------------	-------------

(c)

Block number	Blocks in use
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 2 1 1 1 0 0 1 1 1 1 0 0

0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1	Free blocks
-----------------------------------	-------------

(d)

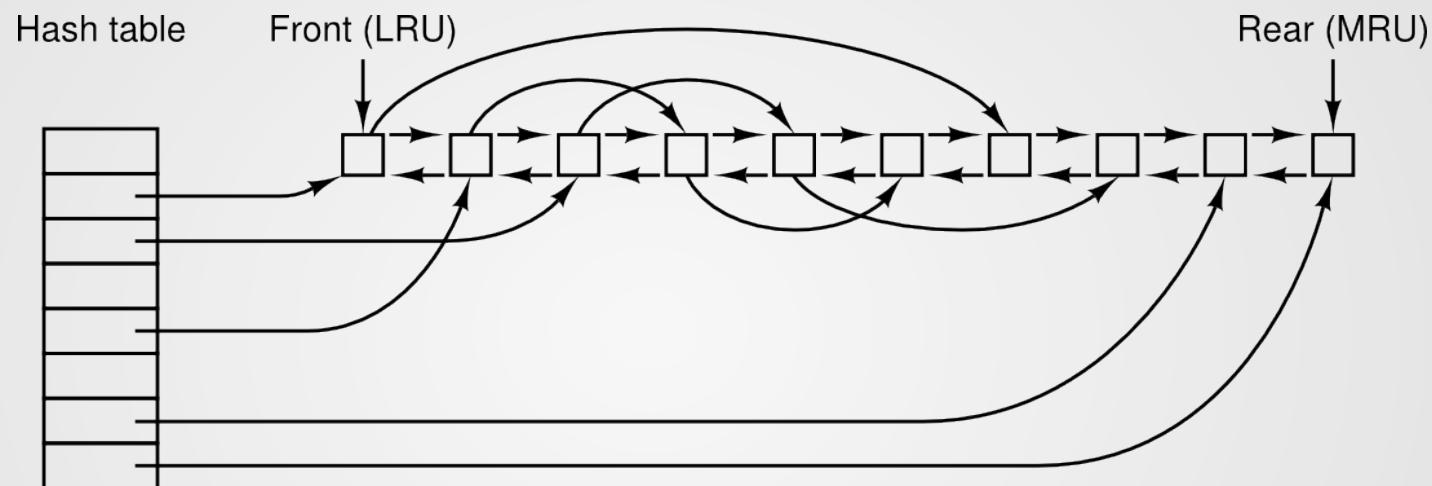
- sui **riferimenti agli i-node**.

Journaling

- **strategia:**
 - le operazioni sui meta-dati sono preliminarmente appuntate in un log che viene poi ripulito a posteriori (subito dopo);
 - in caso di ripristino da crash: ripetere le operazioni in log;
- **benefici:**
 - maggiore robustezza dei metadati;
 - veloce ripristino da crash/reboot;
- affinché tutto funzioni bisogna operare con operazioni **idempotenti**.

Cache del disco

- Per migliorare le prestazioni dei dischi si fa spesso uso di una **cache del disco** (o **buffer cache**):
 - struttura basata su **tabelle hash**;



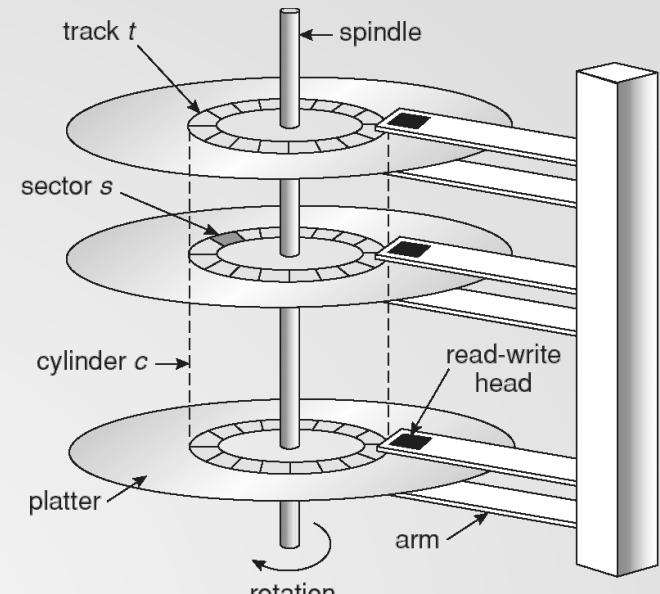
- strategie di gestione:
 - **LRU modificata**;
 - **free-behind & read-ahead**;
- scrittura: **sincrona vs. asincrona**.

Cosa usano i nostri Sistemi Operativi?

- **Windows:**
 - exFAT su unità removibili;
 - NTFS su dischi fissi:
 - file-system moderno, molto complesso;
 - journaling, cifratura, compressione, copia shadow (CoW), dischi multipli (RAID) ...
- **Linux:**
 - **ext-4:**
 - journaling, allocazione efficiente;
 - **BTRFS:**
 - recente file-system evoluto
 - journaling, check-sum dati e metadati, compressione, volumi, clonazione (CoW), dischi multipli (RAID), ...
- **MacOS:** HFS+ recentemente soppiantato da APFS.

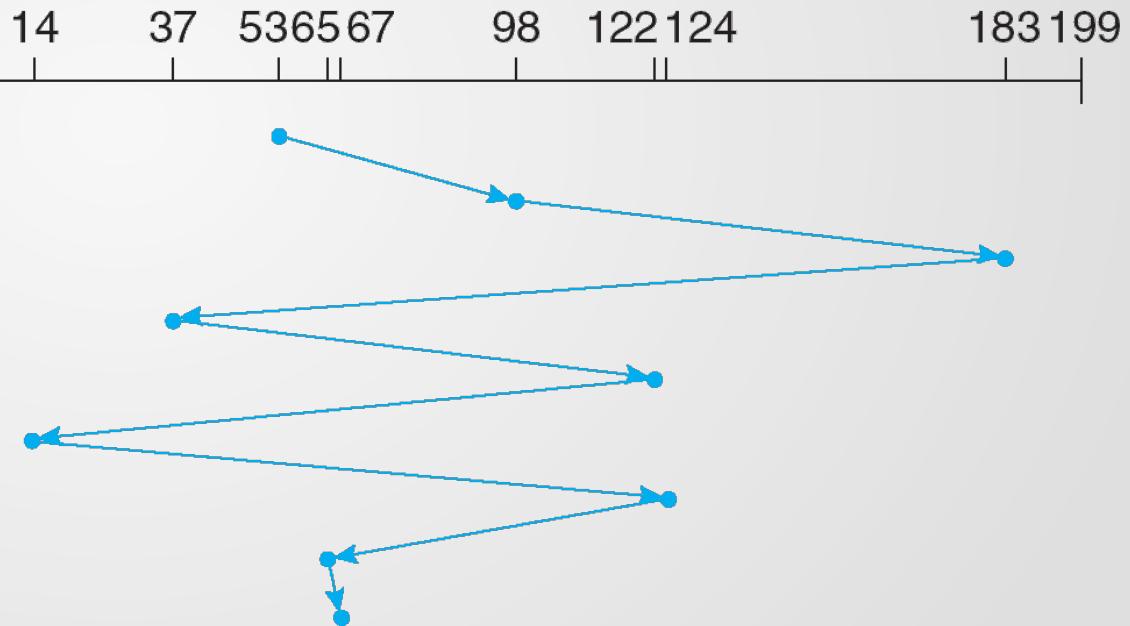
Scheduling del disco

- **Obiettivi:**
 - massimizzare il numero di richieste soddisfatte in una unità di tempo (**throughput**);
 - minizzare il **tempo medio di accesso**;
- in un sistema, soprattutto se multiprogrammato, si vengono a creare varie richieste di I/O su disco che però, tipicamente, possono essere inviate al controller del disco solo una alla volta. Si crea quindi una **coda di richieste pendenti**;
- Il S.O. può adottare varie **politiche di selezione** della prossima richiesta da mandare;
- si può ottimizzare per:
 - **tempo di posizionamento** (seek-time);
 - **latenza di rotazione**.



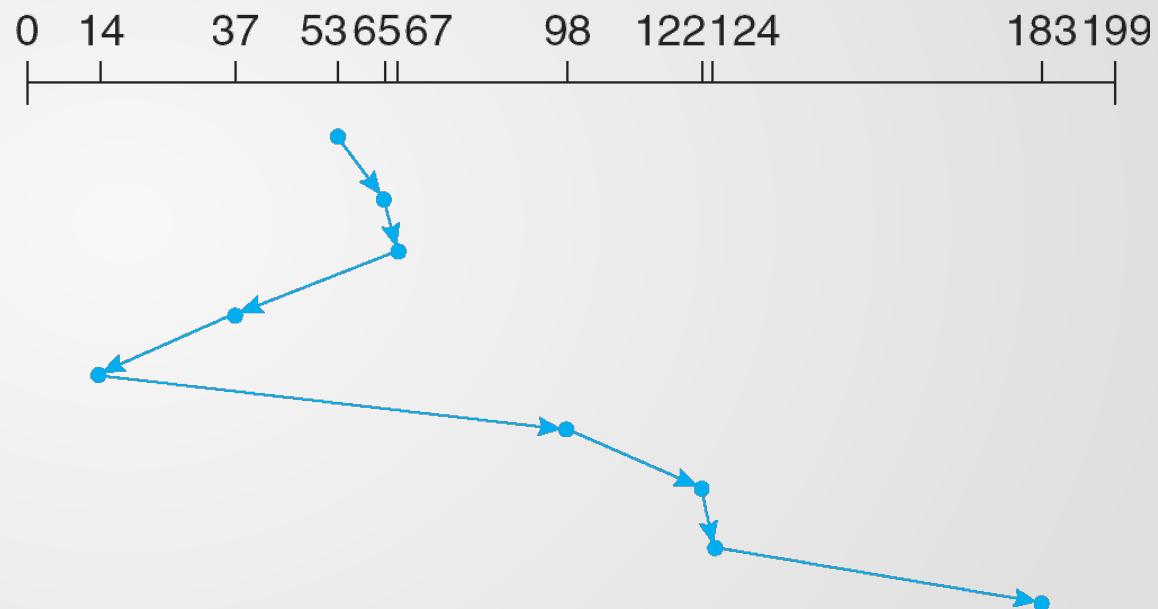
Ottimizzare il seek-time

- Vedremo varie politiche di scheduling su uno specifico **esempio**:
 - lista delle richieste in ordine di arrivo e per # di cilindro:
98, 183, 37, 122, 14, 124, 65, 67
 - posizione iniziale della testina: cilindro **53**
- **First Come First Served (FCFS)**:
 - distanza totale percorsa: 640 tracce;
 - **semplice** da realizzare;
 - **equo**;
 - **inefficiente**;



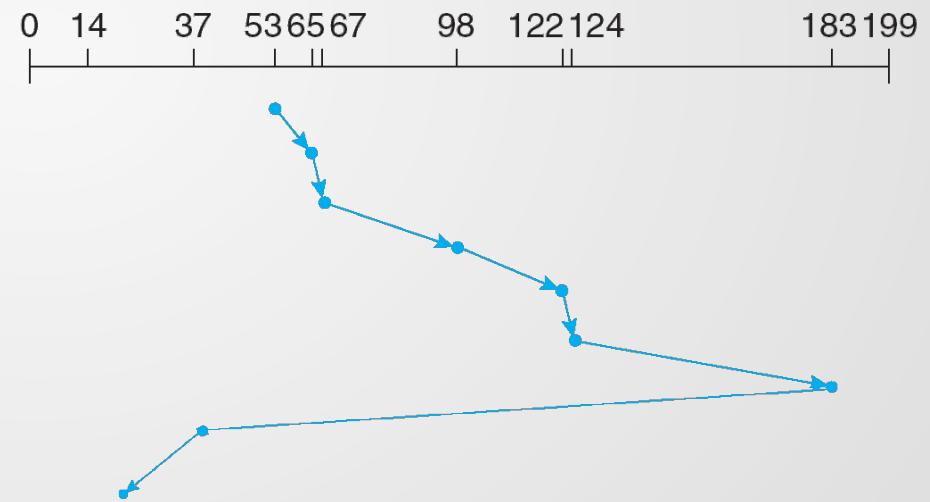
Ottimizzare il seek-time

- **esempio:** coda **98, 183, 37, 122, 14, 124, 65, 67**; cilindro iniziale **53**;
- **Shortest Seek Time First (SSTF):**
 - ordine usato: **65, 67, 37, 14, 98, 122, 124, 183**;
 - distanza totale percorsa: **236 tracce**;
 - **buone prestazioni**;
 - **non equo (*starvation*)**;



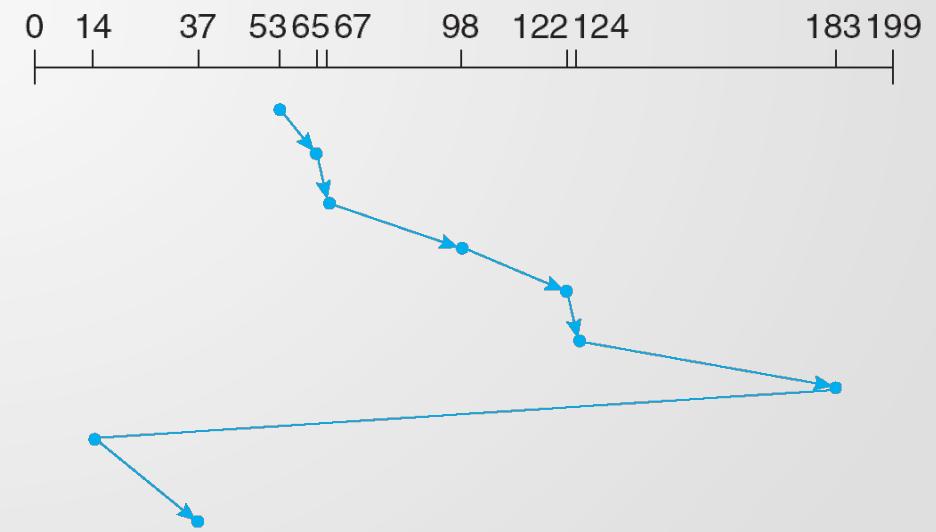
Ottimizzare il seek-time

- **esempio:** coda **98, 183, 37, 122, 14, 124, 65, 67**; cilindro iniziale **53**;
- **Scheduling per scansione** (algoritmo dell'ascensore):
 - mantiene un verso fino all'ultima richiesta in tale direzione
 - ordine usato (inizio UP): **65, 67, 98, 122, 124, 183, 37, 14**
 - distanza totale: 299 tracce (più di SSTF)
 - scansione uniforme;
 - **garantisce comunque una attesa massima per ogni richiesta**



Ottimizzare il seek-time

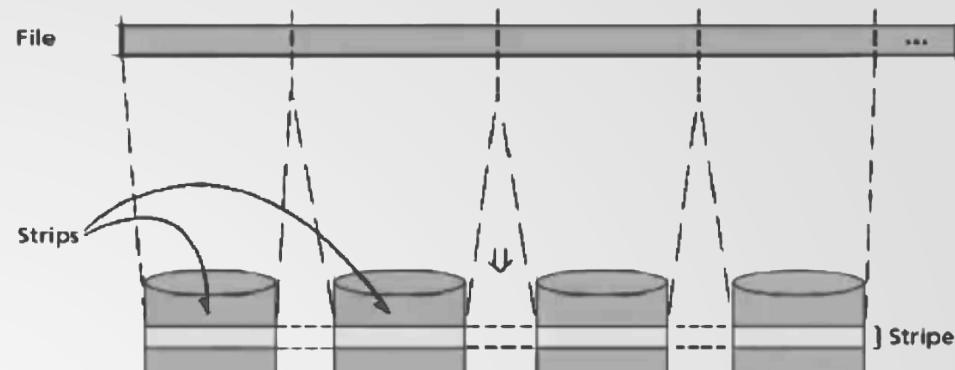
- **esempio:** coda **98, 183, 37, 122, 14, 124, 65, 67**; cilindro iniziale **53**;
- **Scheduling per scansione circolare:**
 - considera le posizioni come collegate in **modo circolare**: arrivato alla fine del disco torna sul primo cilindro senza servire alcuna richiesta;
 - ordine usato: **65, 67, 98, 122, 124, 183, 14, 37**
 - garantisce un **tempo medio di attesa più basso** in presenza di tante richieste.



- **Cosa scegliere?**
 - scansione circolare ad alto carico;
 - scansione o SSTF a basso carico.

Sistemi RAID

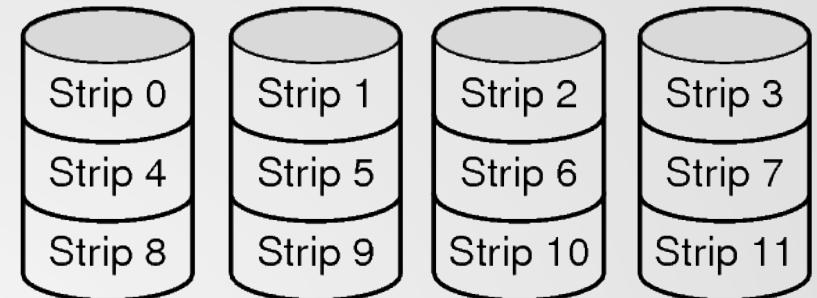
- Un altro modo per aumentare le **prestazioni** è sfruttare il **parallelismo** anche per l'I/O su disco:
 - servono più **dischi indipendenti**;
 - si suddividono i dati relativi ad una unità logica (un file, in generale un volume) su più dischi: **striping**;
 - suddivisione **trasparente all'utente**;
- problema: aumenta la probabilità che si **verifichi un guasto** sul volume logico RAID;
 - soluzione: aggiungiamo ridondanza per ottenere migliore **affidabilità**;
 - sostituzione automatica: dischi **spare**;
- **Redundant Array of Inexpensive Disks (RAID)**;
 - noti anche come **Redundant Array of Independent Disks**;
- vedremo vari schemi di gestione che bilanciano questi due aspetti;
 - **livelli RAID**;
- via **hardware** (trasparente al S.O.) o via **software** (con carico sulla CPU).



Sistemi RAID

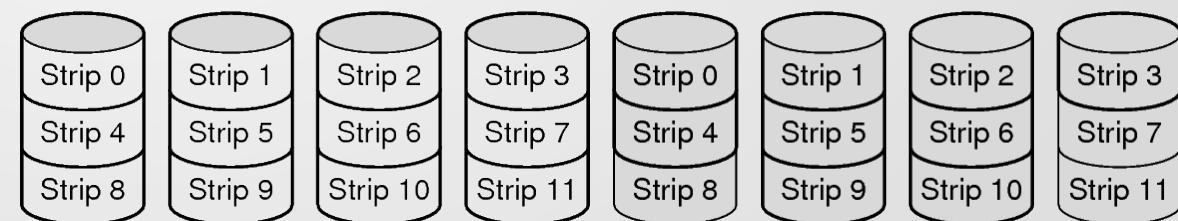
- **RAID 0 (*striping*):**

- fa ***striping*** in modalità round-robin;
- semplice con **prestazioni ottimali** con letture di grandi volumi;
- niente ridondanza:
maggiori vulnerabilità;



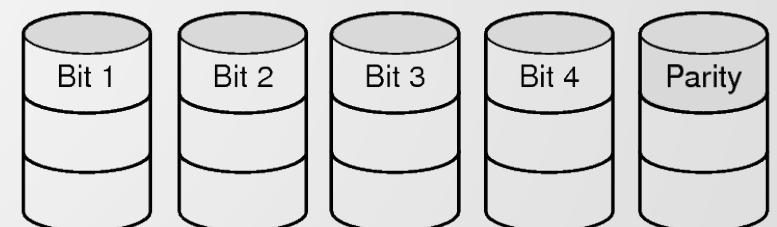
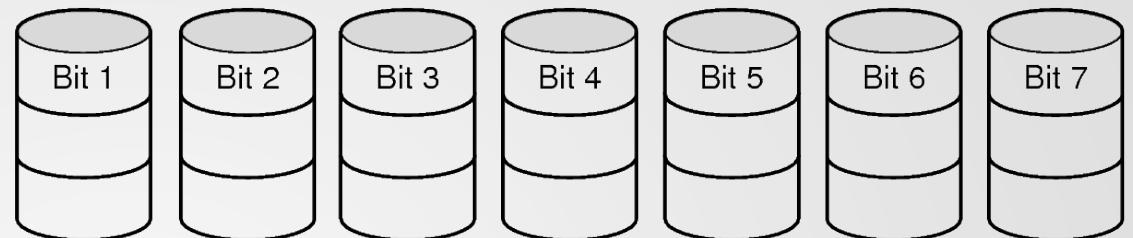
- **RAID 1 (*mirroring*):**

- gli eventuali stripe vengono anche duplicati (***mirroring***);
- può anche essere usato senza striping;
- raddoppio prestazioni in lettura;
- migliore ***fault tolerance***;
- alto **overhead** di storage;



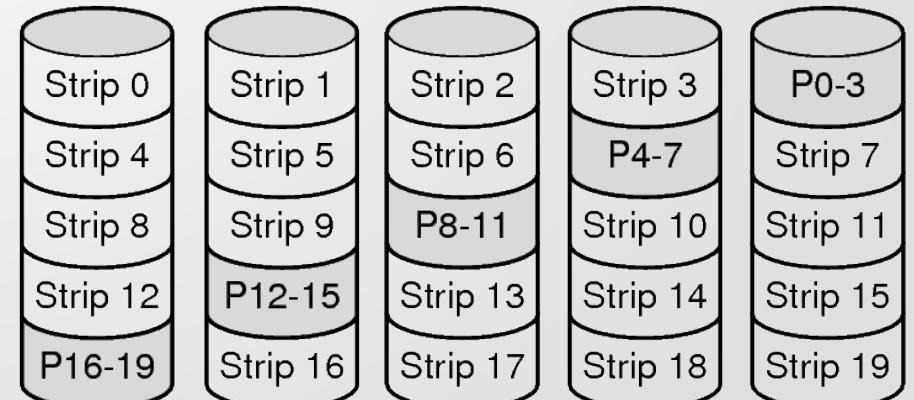
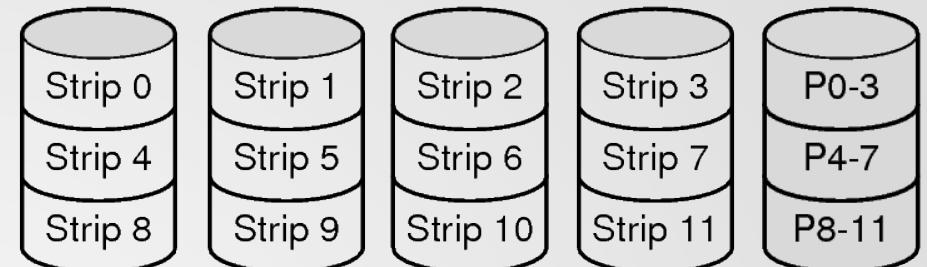
Sistemi RAID

- **RAID 2** (*striping a livello di bit con ECC*):
 - lavora sulle parole applicando un codice di correzione degli errori – ECC (tipo **codice di Hamming** per singoli bit di errore);
 - esempio: 4 bit dati + 3 bit ridondanza;
 - ottima **fault tollerance**;
 - serve **sincronia** nelle rotazioni dei dischi;
- **RAID 3** (*striping a livello di bit con bit di parità*):
 - usa un solo disco con raccolti i singoli **bit di parità**;
 - in realtà permette anche di **recuperare** i dati e offre la stessa capacità di fault tollerance del RAID 2;
 - serve ancora sincronia;
- fare striping a livello di bit è comunque pesante se non gestito a livello hardware;



Sistemi RAID

- **RAID 4** (*striping a livello di blocchi con XOR sull'ultimo disco*):
 - basato sullo **striping a blocchi**;
 - disco extra = **XOR** degli strip;
 - non necessità sincronia;
 - ottima fault tollerance;
 - **aggiornamento** lento in caso di modifica di un blocco?
 - **ottimizzazione**: il nuovo blocco di parità si può calcolare dal blocco sovrascritto e dal vecchio blocco di parità;
- **RAID 5** (*striping a livello di blocchi ma con informazioni di parità distribuite*):
 - il blocchi di parità del RAID 4 vengono distribuiti su tutti i dischi;
 - di fatti sostituisce il RAID 4.



Solid State Disk (SSD)

- I dispositivi basati su memorie flash (**tecnologia NAND**):
 - **lettura molto più veloci** delle scritture;
 - un blocco si deve **cancellare prima** di poter essere riscritto;
 - il numero di cancellature è **limitato** per ogni blocco;
 - **blocco** (unità di cancellazione) composto da **pagine** (unità di allocazione);
 - le letture/scritture sono basata su pagine;
- i **file-system** classici basati su principi diversi;
 - file-system ad-hoc: **Flash-Friendly File System** (F2FS), log-based fs, ...;
 - controller che rimappano i blocchi tramite un **Flash Translation Layer**;
- **garbage collection** e operazione **TRIM**:
 - **degrado** delle prestazioni nel tempo se non impiegato;
 - richiede **adeguamento** da parte del S.O.