Parte V: Gestione della memoria di massa
Parte VI: File System

- Memoria di massa (cap. 11)
- Interfaccia del File System (cap. 13)
- Realizzazione del File System (cap. 14)

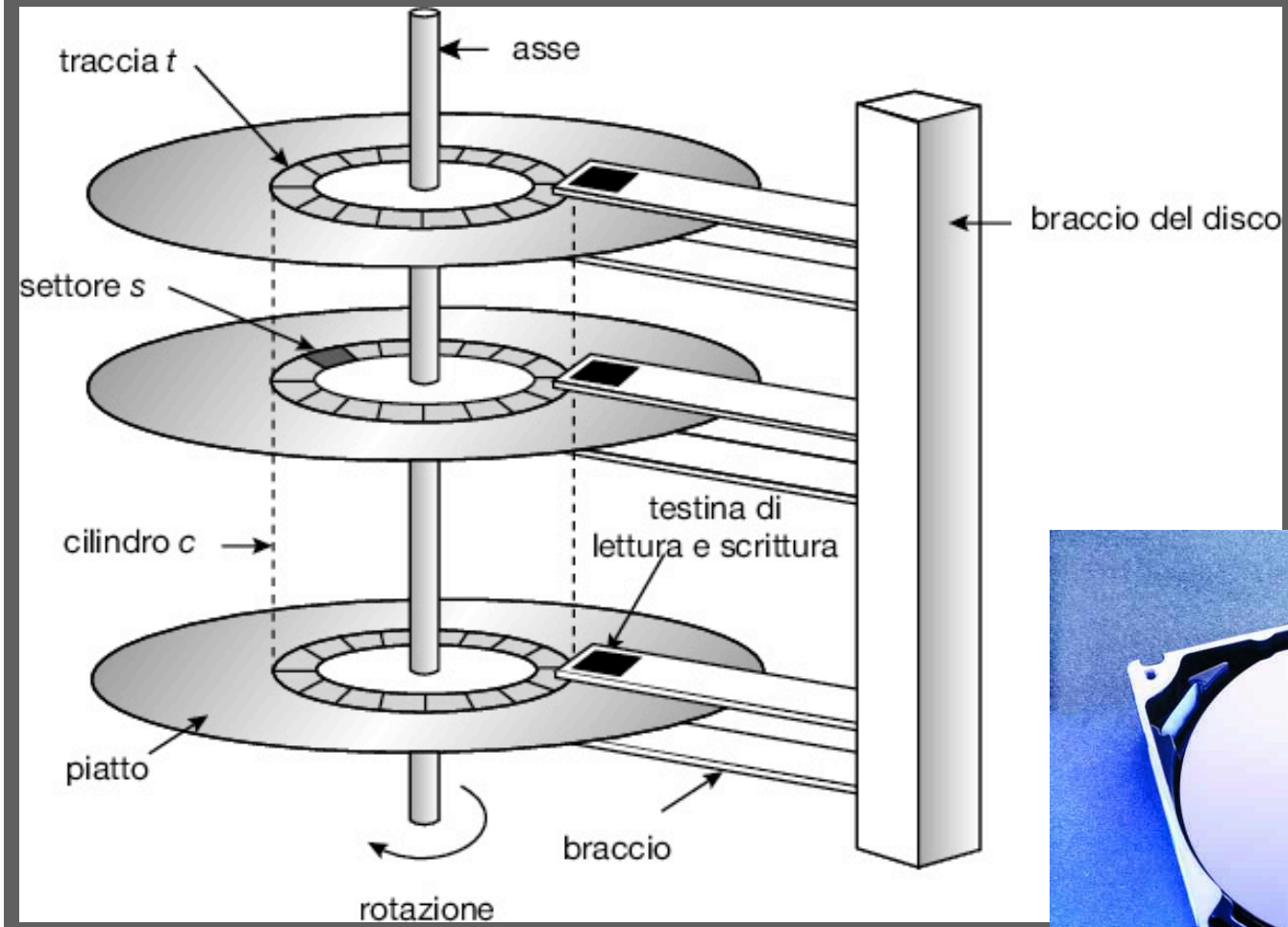
11 Memoria di massa (la memoria secondaria)

- Struttura del disco rigido (o hard disk)
- Scheduling del disco rigido
- Gestione dell'area di swap
- Sistemi RAID
- Memorie a stato solido

11.1.1 Struttura del disco rigido

- Un HD è composto da una serie di "dischi" o **piatti** sovrapposti, di diametro tra 4,5 e 9 cm.
- ogni piatto è suddiviso in una serie di **traccie** circolari concentriche
- ogni traccia è suddivisa in una serie di **settori**
- L'insieme delle traccie nella stessa posizione sui diversi piatti prende il nome di **cilindro**
- Un "braccio del disco" supporta una testina di lettura/ scrittura per ogni piatto: le testine si muovono tutte insieme spostandosi ciascuna sui vari settori del piatto corrispondente (un po' come il braccio di un giradischi)

11.1.1 Struttura del disco rigido



- Schema e hard disk a testine mobili (fig. 11.1, 11.2)

11.1.1 Struttura del disco rigido

- I settori del disco sono l'unità minima di memorizzazione delle informazioni. Fino al 2010 avevano una dimensione standard di 512 byte, ma di recente molti costruttori hanno incominciato a usare settori fino a 4 KB. **Ogni settore memorizza un blocco di dati.**

- I piatti dell'HD ruotano tutti insieme rispetto al loro asse, ad una velocità che a seconda del modello va da 5400 a 15000 RPM (rounds per minute), cioè fino a 250 giri al secondo.

- Ad ogni piatto è associata una testina di lettura/scrittura dei settori, che sfiora la superficie del piatto ad una distanza di pochi micron

11.1.1 Struttura del disco rigido

- Una testina può leggere/scrivere un settore solo quando questo si trova esattamente sotto la testina stessa.
- Dunque, il tempo di accesso ad un settore del disco dipende da due componenti principali
  - **Seek time (tempo di posizionamento)**: il tempo impiegato per spostare la testina sullla traccia che contiene il settore da leggere/ scrivere.
  - **Rotational latency (latenza rotazionale)**: il tempo richiesto perché il piatto che contiene il settore interessato ruoti portando il settore sotto la testina.
- Poiché sono coinvolti elementi meccanici, i tempi di accesso sono sempre dell'ordine di qualche millisecondo

11.1.5 Mappatura degli indirizzi

- Logicamente, un HD può essere visto come un array unidimensionale di **blocchi logici** da 512 (e più di recente 4096) byte: la più piccola unità di trasferimento dati.
- Ogni settore contiene un blocco logico
- L'array unidimensionale di blocchi logici è tradotto in settori del disco sequenzialmente:
  - Il settore 0 è il primo settore della traccia più esterna del primo piatto (di solito il più in alto o il più in basso della pila)
  - Poi si numerano consecutivamente gli altri settori della traccia, fino ai settori delle tracce più interne, e si procede con la numerazione allo stesso modo negli altri piatti.

11.1.5 Mappatura degli indirizzi

- In pratica, la mappatura *blocco logico* → *settore del disco* è resa complicata dai difetti di fabbricazione e dalla diversa lunghezze delle tracce
- Infatti, i dischi possono avere settori difettosi che vengono nascosti dal meccanismo di mappatura tra il numero dei blocchi logici e i settori sul disco
- Inoltre, il numero di settori per traccia non è costante: più una traccia è lontana dal centro del disco e più è lunga, e quindi maggiore è il numero di settori che può contenere (fino al 40% in più di una traccia interna)

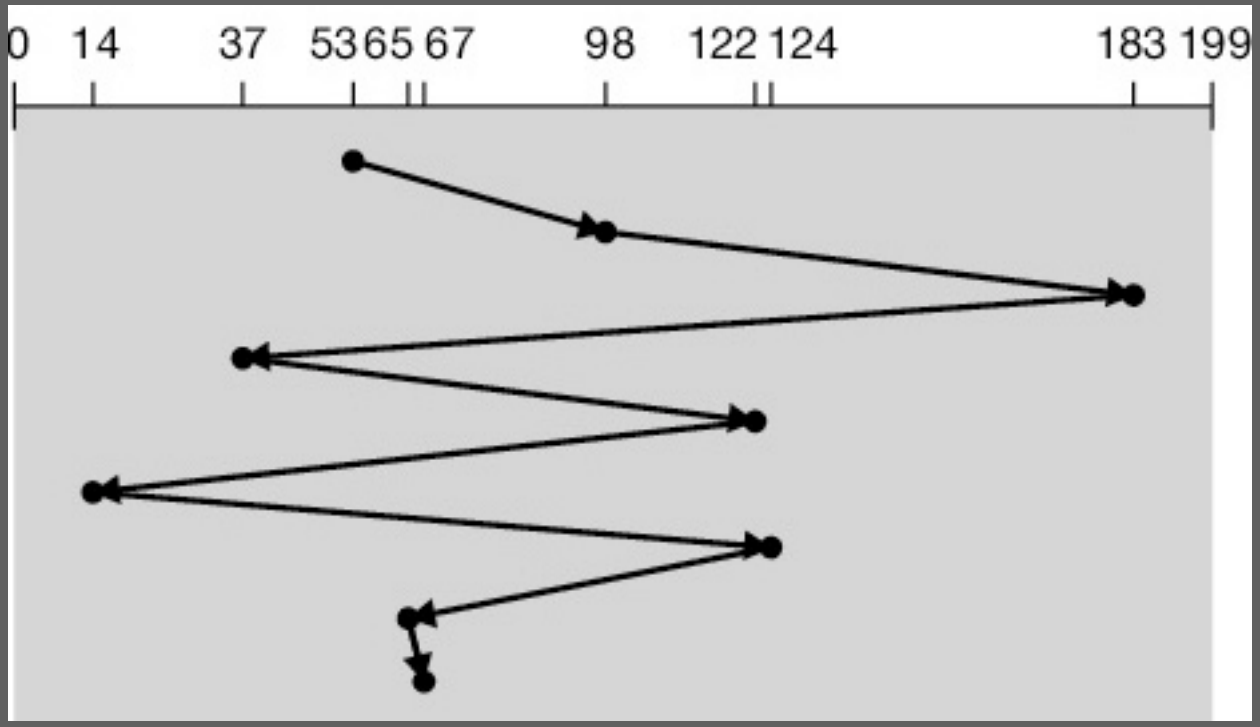
11.2 Scheduling dei dischi rigidi

- In generale, il SO riceve da parte dei processi molte richieste di accesso a dati memorizzati sul disco, e deve cercare di organizzare il trasferimento delle informazioni in modo da ottimizzare le prestazioni di accesso al disco
- **Il SO non può influenzare la latenza rotazionale** del disco che in media sarà pari a ½ del tempo necessario a compiere una rotazione completa.
- **Però il SO può cercare di minimizzare il seek time medio complessivo**, riordinando le richieste in attesa di essere servite in modo che le testine si debbano muovere il meno possibile.

11.2 Scheduling dei dischi rigidi

- Esistono diversi algoritmi di scheduling delle richieste di I/O del disco. Qui ne vediamo solo due, utilizzando come esempio una sequenza di richieste di lettura/scrittura comprese fra la traccia 0 e la traccia 199.
- 98, 183, 37, 122, 14, 124, 65, 67
- Notiamo che le tracce potrebbero anche trovarsi su piatti diversi, dato che comunque i piatti ruotano tutti assieme e le testine dei piatti si muovono anch'esse tutte insieme.
- Ma per seplicità possiamo anche supporre che esista un solo piatto e che la sua testina sia posizionata inizialmente sulla traccia (o sul cilindro composto da un'unica traccia) n. 53

11.2.1 Scheduling FCFS

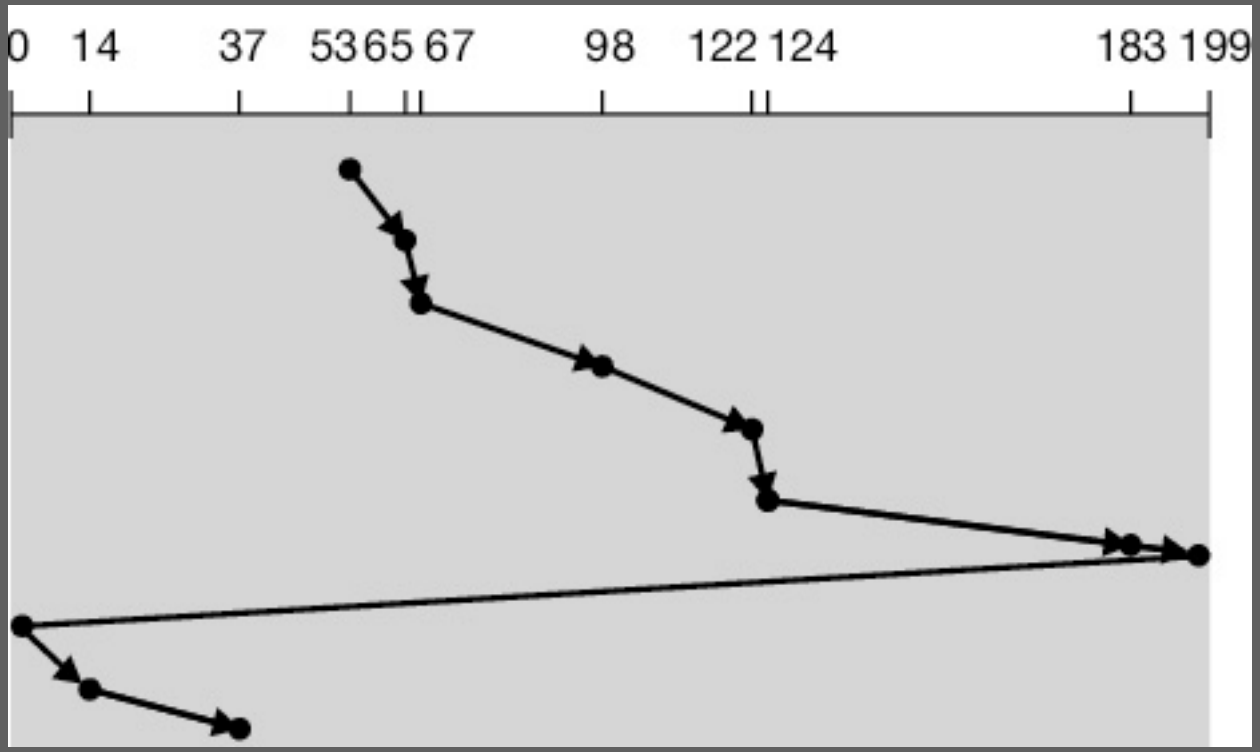


- Coda delle richieste: 98, 183, 37, 122, 14, 124, 65, 67
- In tutto la testina attraversa 640 tracce. Invece che “122 – 14 – 124” era meglio fare “122 – 124 – 14” (Fig. 11.6)

11.2.3 C-SCAN (Circular SCAN)

- Fornisce un tempo di attesa per le varie richieste più uniforme di altri algoritmi, anche se non riesce a garantire un tempo medio di attesa minimo.
- La testina si muove da un estremo all'altro del piatto, servendo le richieste.
- Quando raggiunge l'estremità del piatto, torna immediatamente all'inizio senza servire richieste.
- In pratica, tratta i settori/cilindri come una lista circolare.

11.2.3 C-SCAN



- Coda delle richieste: 98, 183, 37, 122, 14, 124, 65, 67
- La testina attraversa 183 tracce + 200 tracce per tornare indietro, ma questo ritorno richiede poco tempo, perché la testina non deve mai fermarsi e ripartire (Fig. 11.8)

11.5.1 Formattazione del disco

- Per poter essere usato, un disco ha sempre bisogno di subire un processo di **formattazione a basso livello** (o **formattazione fisica**)
- Questa formattazione (fatta di solito dal costruttore dell'HD) permette di associare ad ogni settore il suo numero, e di prevedere uno spazio per inserire un codice di correzione degli errori, usato in ogni operazione di I/O su quel settore
- In fase di formattazione fisica, è anche possibile scegliere la dimensione dei blocchi fisici (512 / 4096 byte per settore)

11.5.1 Formattazione dell'HD

- Il SO sottopone poi l'HD ad una **formattazione logica**, necessaria per creare e gestire sull'HD il File System del SO.
- Il SO crea quindi la lista dei blocchi liberi (secondo lo schema adottato), e una directory iniziale, da cui si dipartiranno tutte le altre
- Sull'HD vengono poi riservate le aree che dovranno essere gestite direttamente dal SO:
  - il boot block (blocco di avviamento)
  - l'area che contiene gli attributi dei file (ad esempio, gli index-node Unix o la MFT di Windows)

11.5.1 Il blocco di avviamento (boot block)

- Il Boot Block contiene il codice necessario per far partire il sistema operativo.
- All'accensione, un piccolo programma contenuto in ROM istruisce il disk controller in modo da trasferire il contenuto del Boot Block in RAM.
- Il controllo è trasferito al codice del Boot Block, che si occupa di far partire l'intero SO prelevando in codice dal disco stesso.

11.6 Gestione dell'area di Swap

- (Nota: sul testo l'area di swap è chiamata *area d'avvicendamento*. Suggerisco di evitare questa traduzione un po' goffa)
- Durante la formattazione logica dell'HD, il SO riserva a se stesso uno spazio da usare come area di swap.
- Nel caso più semplice, l'area di Swap può essere un file molto grande all'interno del FS.
- Windows usa appunto questa soluzione, e lo swap file si chiama **pagefile.sys**. Potete cambiargli dimensione, ad esempio riducendola, se avete molta RAM a disposizione. In questo modo recuperate spazio sull'hard disk.

11.6 Gestione dell'area di Swap

- Alternativamente, una porzione specifica dell'HD (più propriamente detta "partizione") può essere riservata per l'area di Swap.
- Questa partizione non viene trattata allo stesso modo del FS, ma vengono usate strategie di allocazione diverse per migliorare al massimo la velocità di accesso.
- Ad esempio, l'allocazione dei blocchi alle pagine swappate in memoria secondaria può essere contigua, in modo da non dover gestire un meccanismo di ricerca dei blocchi liberi.

11.6 Gestione dell'area di Swap

- Perché tutto funzioni adeguatamente, è altamente preferibile sovradimensionare l'area di Swap, in modo che il SO trovi sempre in fretta un'area libera per lo swap di pagine e segmenti.
  - Ad esempio, in Solaris si consiglia di usare un area di Swap pari alla differenza tra lo spazio di indirizzamento logico e quello fisico
  - Linux suggerisce di usare un'area di swap di dimensione doppia a quella della RAM disponibile.
- Nei sistemi con più dischi, si può prevedere un'area di swap per ciascun disco, in modo da poterle usare contemporaneamente, bilanciando il carico di lavoro

11.8 Sistemi RAID

- Un HD (ma anche un SSD) è più lento del processore e della memoria primaria.
- E il guasto di un hard disk è potenzialmente il più dannoso: se si guasta e non c'è un back-up, **tutti i dati che contiene** vanno persi (o per lo meno non saranno disponibili per il tempo necessario alla riparazione)
- Un sistema **RAID** è un modo di configurare la memoria secondaria che aumenta le prestazioni degli hard disk e la loro affidabilità.
- Queste "architetture" di memoria secondaria, utili in qualsiasi settore, sono addirittura fondamentali in quei contesti in cui il servizio fornito non può mai venir meno, ad esempio in campo finanziario e bancario.

11.8 Sistemi RAID

- Il concetto di dispositivo RAID è stato introdotto nel 1988 da Patterson, Gibson e Katz, e si è velocemente affermato a livello commerciale.
- L'acronimo inizialmente coniato da Patterson stava per **Redundant Array of Inexpensive Disks,** presto però ridefinito dalle varie compagnie produttrici di sistemi RAID in **Redundant Array of Independent Disks**
- In modo un po' simile alla contrapposizione RISC / CISC, fu definita anche la controparte dei sistemi RAID, indicata con l'acronimo **SLED**: **Single Large Expensive Disk**.

11.8 Sistemi RAID

- Un sistema RAID è composto da un insieme di hard disk (un *disk array*) ma viene visto dal sistema operativo che lo usa come un normale disco singolo (dovremmo dire: come uno SLED), ma **più veloce** ed **affidabile** di uno SLED.
- Il controller del RAID gestisce i dischi (secondo criteri che vedremo più avanti) in modo che il SO li veda come un unico dispositivo di memorizzazione.
- La possibilità di usare un sistema RAID come se fosse un normale hard disk fa si che non siano necessari cambiamenti software nel sistema operativo che deve usare il RAID (il che è ovviamente un vantaggio, specie per i system administrators del sistema)

11.8 Sistemi RAID

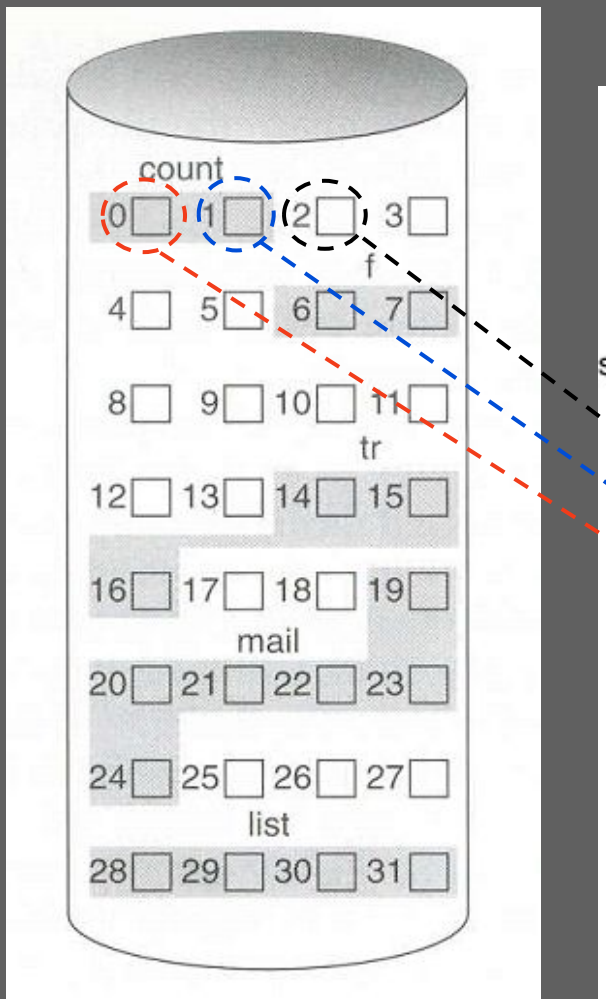
- le idee di fondo di un sistema RAID sono due:

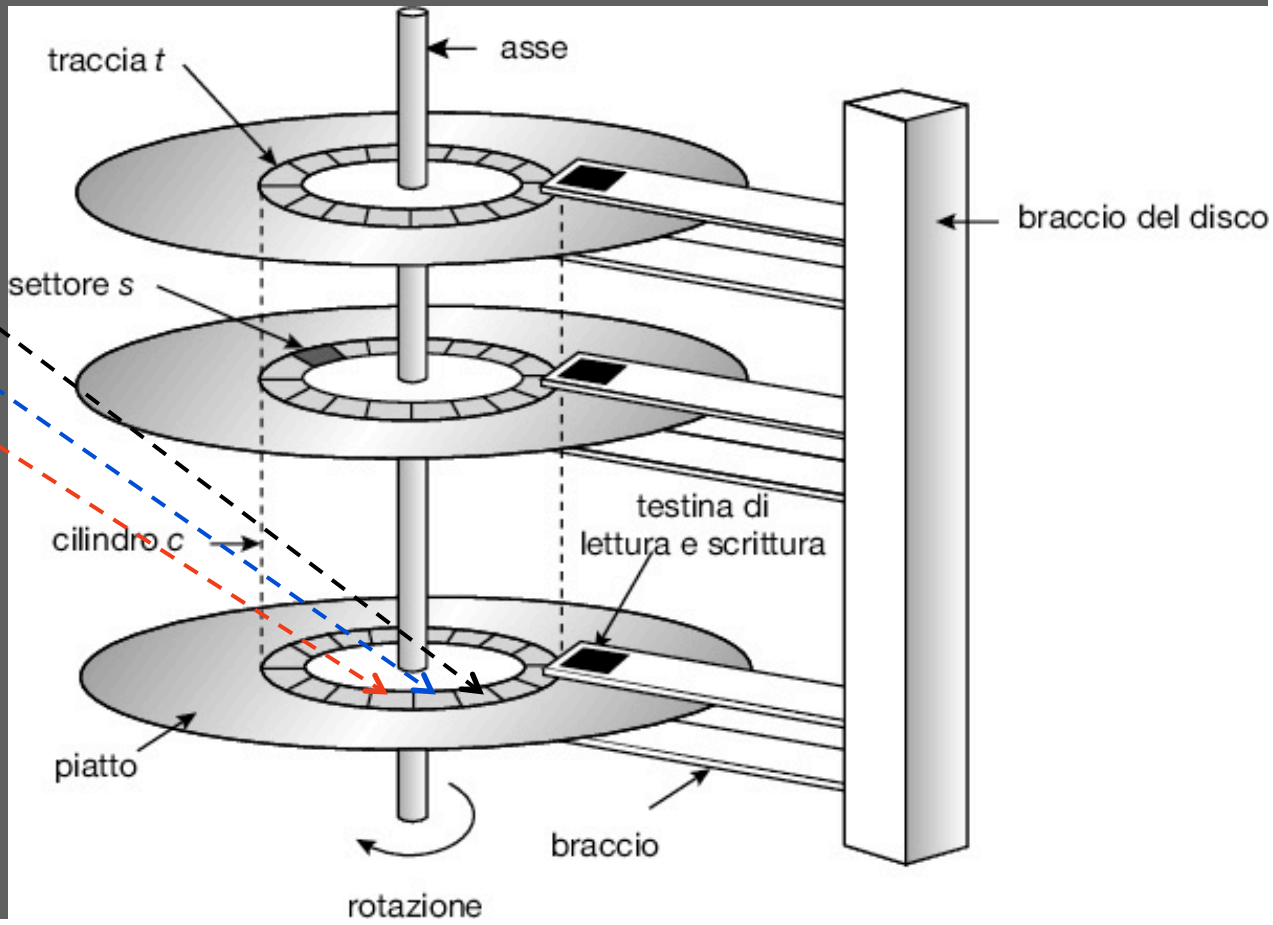
1. **distribuire l'informazione memorizzata su più dischi**, in modo da parallelizzare una parte delle operazioni di accesso ai dati e migliorare le prestazioni.
2. **Duplicare l'informazione memorizzata su più dischi**, così in caso di guasto di un disco è possibile comunque mantenere funzionante il sistema, recuperando in qualche modo l'informazione memorizzata sul disco guasto.

- Differenti schemi sono stati proposti per realizzare i punti 1 e 2, a cui corrispondono diversi livelli di sistemi RAID: dal livello 0 al livello 6. (N.B. da qui in poi ci discostiamo un po' dalla trattazione fatta sul libro di testo)

Ricordiamo dalla sezione 11.1.1

- Il SO vede il disco come un array di blocchi, che sono mappati sui vari settori dell'HD dal controller del disco.



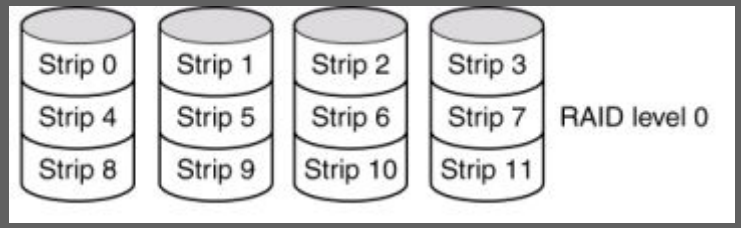


RAID livello 0 (block striping)

- I sistemi RAID di livello 0 non sono in senso stretto dei sistemi RAID, in quanto non c'è duplicazione dei dati.
- In un RAID livello 0, il disco virtuale (ossia ciò che viene visto dal SO: un insieme di blocchi logici numerati consecutivamente) viene mappato dalla logica del RAID sui settori dei vari dischi di cui è composto il sistema suddividendo i blocchi logici del disco virtuale in **strips** (strisce) di k blocchi consecutivi ciascuna.
- Quindi, lo strip 0 contiene i blocchi logici da 0 a k - 1, lo strip 1 contiene i blocchi logici da k a $2k$ - 1, e così via.

RAID livello 0 (block striping)

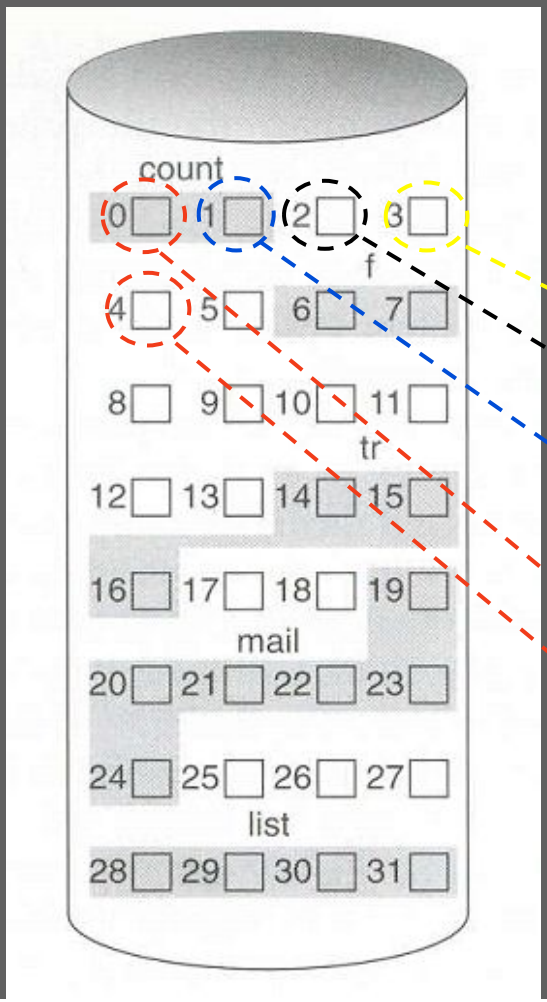
- Il RAID controller suddivide poi gli strip tra i dischi del sistema secondo la formula:
  numero-strip MOD *dischi-nel-sistema*.
- Se sono disponibili 4 dischi, il disco 0 conterrà gli strip 0, 4, 8,... il disco 1 conterrà gli strip 1, 5, 9,..., e così via (Tanenbaum, Fig. 2-23a)



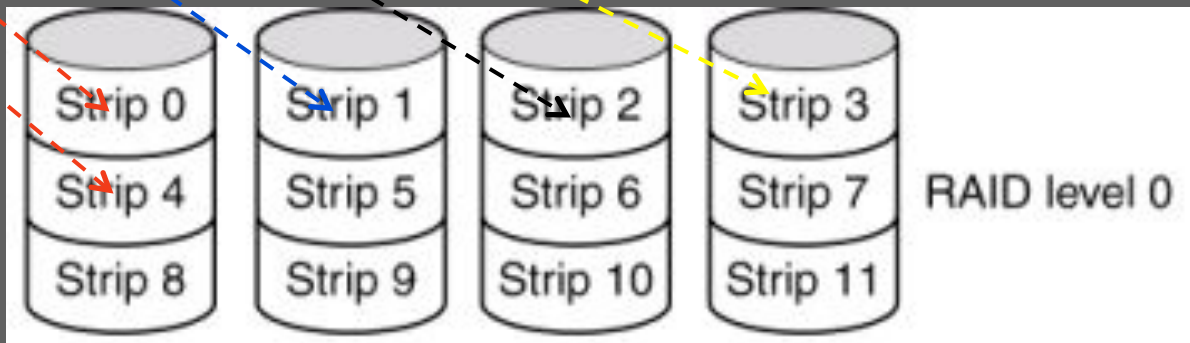
- Questa tecnica è nota come **striping**, e in realtà era già adottata prima dell'introduzione del concetto di RAID.
- I sistemi RAID offrono comunque il livello 0, nel caso in cui si vogliano solo massimizzare le prestazioni e la capacità del sistema senza aumentarne l'affidabilità.

RAID livello 0 (block striping)

Notate: se $k = 1$, ogni strip contiene un blocco. Il blocco 0 è mappato sul primo settore del primo disco, il blocco 1 è mappato sul primo settore del secondo disco, e così via.



Il blocco 4 è mappato sul secondo settore del primo disco, il blocco 5 è mappato sul secondo settore del secondo disco, ecc.



RAID livello 0 (block striping)

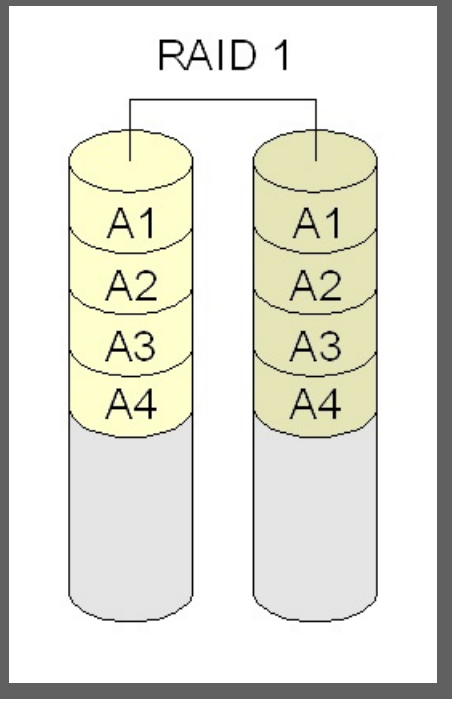
- Perché le prestazioni di un sistema RAID di livello 0 sono migliori di un normale disco che memorizza le stesse informazioni?
- Supponiamo che il SO richieda la lettura (o scrittura) di un insieme di dati contenuti in 4 strip consecutivi. Ad esempio, i dati sono contenuti negli strip 4, 5, 6 e 7.
- Il controller RAID suddividerà la richiesta in 4 letture (scritture) eseguite in parallelo sui 4 dischi del sistema. Al contrario, su di un singolo disco, tutti i settori dei quattro strip dovranno essere letti (scritti) in sequenza
- Il SO non si accorge di nulla, ma completerà l'operazione più velocemente che nel caso di un normale disco SLED

RAID livello 0 (block striping)

- Un RAID di livello 0 è tanto più efficiente quanto più le richieste coinvolgono l'accesso a molti strip consecutivi ed è alto il numero di dischi su cui sono suddivisi gli strip.
- Al contrario, richieste di dati contenuti in un unico strip, non ottengono alcun miglioramento di prestazioni rispetto ad un disco SLED.
- inoltre, l'affidabilità di un sistema RAID di livello 0 è inferiore a quella di un semplice disco SLED, perché il RAID è formato di più dischi e il **Mean Time To Failure (MTTF)** non può che essere inferiore.
- Il RAID livello 0 viene usato in quelle applicazioni in cui sono necessarie alte prestazioni, senza particolari problemi di affidabilità (e.g., audio e video streaming)

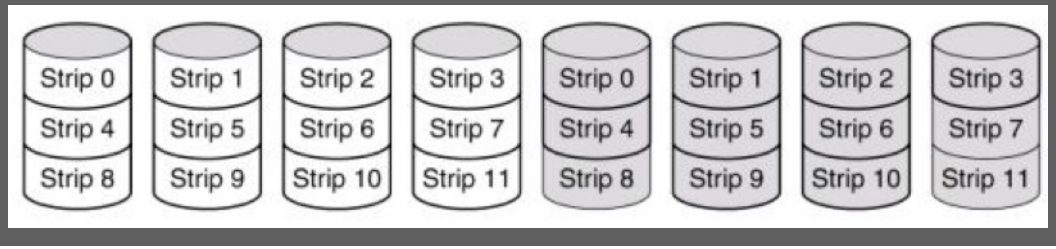
RAID livello 1 (mirroring)

- Nel RAID di livello 1, per ogni disco di dati D viene usato un secondo disco di **mirroring** che contiene una copia esatta del disco D.
- La configurazione più semplice è quindi quella che usa due dischi: uno conterrà i dati e il secondo sarà una copia perfetta del primo.
- Questa configurazione fornisce una maggiore affidabilità, perché se un disco si rompe ne è disponibile una copia identica. Particolarmente interessante è però la combinazione dei livelli 0 e 1, che fornisce i massimi livelli di prestazioni e affidabilità (Fig. Wikipedia)



RAID livello 01 (striping+mirroring)

- Il RAID di livello 01 (detto anche 0+1) usa insieme striping e mirroring: tutti i dati sono suddivisi in strip distribuiti su più dischi (come nel livello 0) e ciascun disco viene poi duplicato (come nel livello 1)
- Quando un disco si rompe, il sistema di controllo del RAID si rivolge al disco di mirror per l'accesso ai dati. Nel frattempo, il disco rotto può essere sostituito (anche con procedure automatiche se è disponibile un ulteriore *spare disk,* su cui verranno copiati i dati contenuti nel "gemello" del disco rotto). (Tanenbaum, Fig. 2-23b)

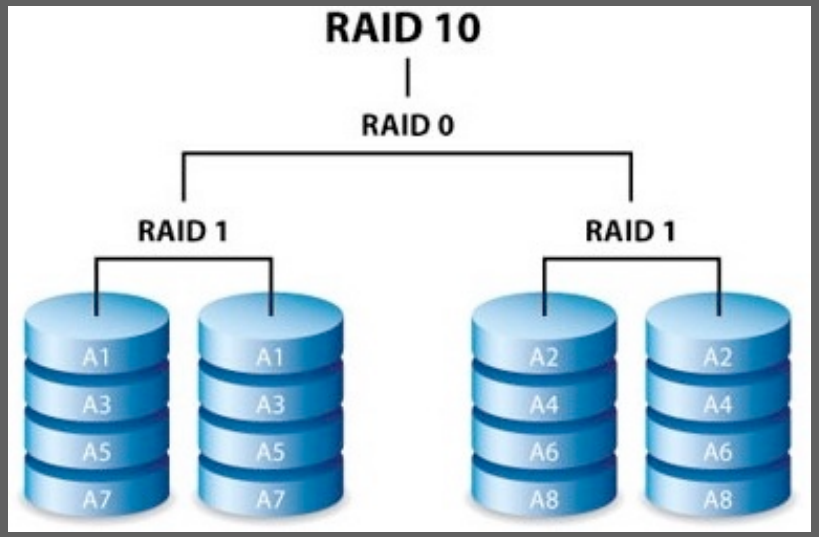


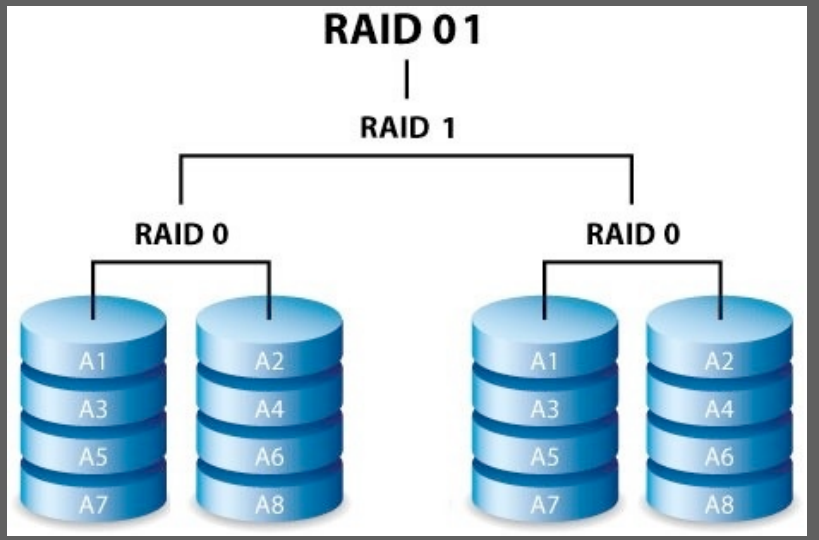
RAID livello 01 (striping+mirroring)

- Il livello 01 è la soluzione RAID più costosa, a parità di capacità di memorizzazione, perché richiede la duplicazione di tutti i dischi.
- E' anche una soluzione tra le più affidabili rispetto ai guasti, e la più efficiente in lettura: anche la lettura di un blocco di dati che coinvolge 5 strip (nel nostro esempio), può essere eseguita con cinque letture in parallelo, usando anche i dischi di mirroring.
- Il RAID livello 01 viene usato dove l'affidabilità è fondamentale, ad esempio per memorizzare dati finanziari e bancari. Quanto maggiore è il numero di dischi coinvolti, tanto maggiori saranno le prestazioni.

RAID livello 10 (mirroring+striping)

- Il livello 10 (detto anche 1+0) è una variante del 01, in cui mirroring e striping vengono fatti in modo inverso al livello 01 (nel livello 10 viene prima fatto il mirroring a coppie dei dischi, e poi lo striping di queste coppie).
- Le prestazioni dei RAID 01 e 10 sono simili, anche se il RAID 10 offre alcuni vantaggi in caso di guasti





Risalire ad un dato perso attraverso la stringa di parità

- Abbiamo *m* stringhe binarie lunghe ciascuna *n* bit. Ogni stringa è memorizzata su un diverso hard disk. Uno dei dischi si rompe. È possibile ricostruire la stringa persa?
- Si se avevamo calcolato e salvato la cosiddetta parità bit a bit delle *m* stringhe così:
- parità =
  stringa-0 EX-OR stringa-1 EX-OR … EX-OR stringa-m.
- Notate che |parità| = |stringa-j|. Se perdiamo stringa-j:
- Stringa-j = stringa-0 EX-OR stringa-1 EX-OR … … EX-OR stringa-m **EX-OR parità**

Risalire ad un dato perso attraverso la stringa di parità

- Vediamo un esempio in cui le stringhe sono 3, hanno lunghezza 8 bit, e viene persa la stringa *a*.

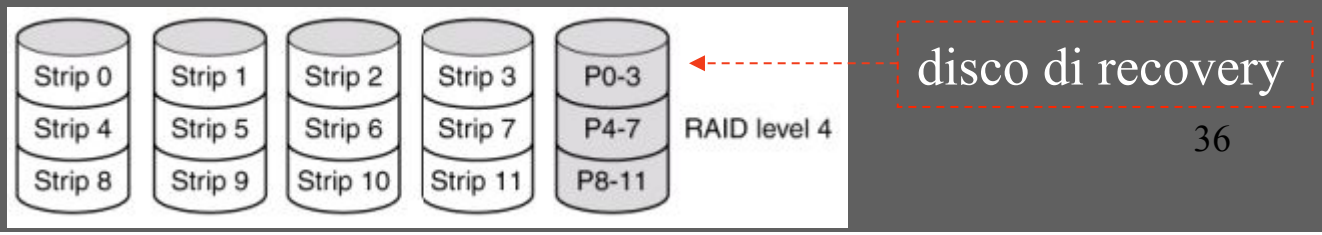
a) 01100011 EX-OR
b) 10101010 EX-OR
c) 11001010 =
p) 00000011

p) 00000011 EX-OR
b) 10101010 EX-OR
c) 11001010 =
a) 01100011

- Notiamo che le stringhe possono avere lunghezza arbitraria (purché uguale per tutte), e dunque una stringa può essere costituita da tutti i bit che costituiscono uno strip di un sistema RAID

RAID livello 4 (striping con parità)

- Il RAID livello 4 usa lo striping a livello di blocchi, e calcola uno strip di parità per poter implementare una eventuale operazione di recovery.
- Un disco memorizza gli strip di parità calcolati con gli strip che stanno nella stessa posizione negli altri dischi.
- Ad esempio, (Tanenbaum, Fig. 2-23e), nel caso di un sistema con 5 dischi, il primo strip del disco di recovery conterrà la parità calcolata usando gli strip 0, 1, 2 e 3, il secondo strip di parità conterrà la parità degli strip 4, 5, 6 e 7, e così via.

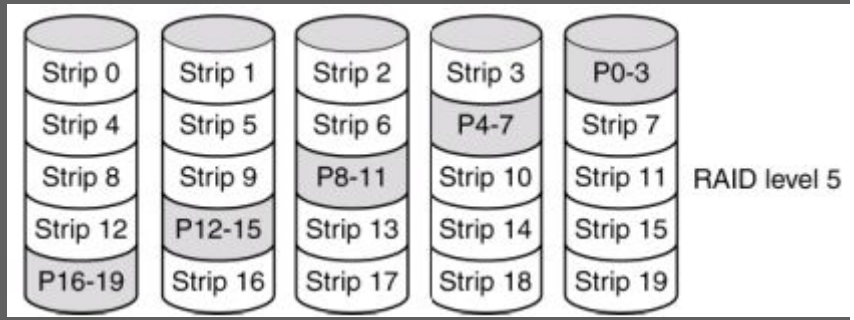


RAID livello 4 (striping con parità)

- Il RAID livello 4 risparmia dischi rispetto al RAID livello 01, e garantisce lo stesso il mantenimento dei dati in caso di guasto di un disco, ma con minore efficienza
- Infatti, se lo strip di un disco viene modificato, vanno letti anche i corrispondenti strip di tutti gli altri dischi per ricalcolare la parità. Oppure, si può leggere il vecchio strip, sottrarlo allo strip di parità, e ricalcolare lo strip di parità usando il nuovo strip
- Ma la cosa più grave è che il disco che ospita gli strip di parità è coinvolto in ogni operazione di scrittura sul RAID, e può facilmente diventare un collo di bottiglia e ha maggiori probabilità di guastarsi

RAID livello 5 (striping con parità distribuita)

- Il RAID livello 5 funziona come il 4, ma riduce il carico sul disco di parità nelle operazioni di scrittura, distribuendo gli strip di parità fra tutti i dischi (Tanenbaum, Fig. 2-23f)
- Questo rende più complessa la ricostruzione del contenuto di un disco che si è rotto, dato che ora i dischi contengono sia strip di dati che strip di parità



- Questo livello fornisce comunque la migliore combinazione in termini di prestazioni, affidabilità, e capacità di memorizzazione, ed è quindi di gran lunga il livello più usato per applicazioni generiche.

RAID livello 6 (striping + doppia parità distribuita)

- L'ultimo livello RAID è il livello 6, in grado di resistere anche al guasto di due dischi contemporaneamente, combinando due livelli di parità, distribuiti sui vari dischi come nel caso del livello 5.
- Per questo sistema è necessario un disco in più del livello 5 per memorizzare la stessa quantità di dati, e un maggiore overhead computazionale.
- Tuttavia, questa soluzione è poco usata perché la rottura contemporanea di due dischi è un evento estremamente raro (l'MTTF di un singolo disco è ormai stimato in anni)

11.1.2 Memorie a Stato Solido

- Negli ultimi anni sono andate sempre più diffondendosi le **memorie a stato solido** basate sulla tecnologia delle **memorie flash** come supporto di memoria secondaria (e infatti spesso vengono semplicemente chiamate memorie flash).
- Queste memorie sono usate per implementare la memoria di massa nei dispositivi mobili, come smart phone e tablet.
- Più di recente vengono anche usate nei computer, ad integrazione del disco rigido. A volte sostituiscono il disco rigido, soprattutto nei portatili di fascia alta, per renderli più leggeri e più veloci (ma anche più costosi).

11.1.2 Memorie a Stato Solido

- Più in dettaglio, le memorie a stato solido usate al posto di un normale hard disk, e montate su opportuni contenitori o schede da inserire nel computer, sono indicate come **SSD: Solid State Disk,** che è ovviamente un termine fuorviante, dato che non cè nessun disco coinvolto…(Fig 11.3: una scheda SSD da 3,5 pollici)
- Mentre quando sono incapsulate in un contenitore rimuovibile attraverso una porta USB vengono chiamate **pen drive** o **pen flash**

11.1.2 Memorie a Stato Solido

- In sostanza, le memorie a stato solido sono memorie permanenti ma riscrivibili. In maniera simile ai settori dell'HD, queste memorie sono organizzate in pagine da 2 a 16 Kbyte, e ogni lettura/scrittura coinvolge l'intera pagina.
- Sono memorie permanenti ma con un limite di circa 100.000 riscritture per pagina (il che è più che sufficiente per qualsiasi applicazione ragionevole)
- Prima di essere riscritta, la pagina deve essere cancellata (appunto con un processo di "flashing", da cui il nome delle memorie). Questo rende le memorie flash molto più lente in scrittura che in lettura.

11.1.2 Memorie a Stato Solido

- Costi:
  - **Hard disk**: circa 0,09$ alGigaByte
    Flash memory: circa 2$ al GigaByte,
    RAM memory: circa 20$÷40$ al GigaByte
- Velocità in lettura: (sia T il tempo di lettura in RAM)
  - **Hard disk**: circa 1000 × T
    Flash memory: circa 4 × T
    RAM memory: T
- Velocità in scrittura: (sia T il tempo di scrittura in RAM)
  - **Hard disk**: circa 1000 × T
    Flash memory: da 10 × T fino a 100 × T
    RAM memory: T

11.1.2 Memorie a Stato Solido

- Se non lo sostituisce, un SSD può complementare l'uso di un hard disk in due modi:
- Come livello di memoria cache permanente collocata tra l'hard disk e la memoria RAM
- Come un secondo hard disk più piccolo ma più veloce, su cui memorizzare (completamente o parzialmente) il sistema operativo, gli applicativi, i file più usati, l'area di swap. Tutti gli altri file verranno memorizzati sull'hard disk.
- Poiché l'SSD è un dispositivo ad accesso completamente diretto (al contrario dell'hard disk), non necessita di una politica di scheduling particolare, e si usa di solito FCFS

Per chi vuole approfondire

- Sezione 11.6: gestione dell'area di swap
- Sezione 11.7: connessione dei dispositivi di memorizzazione