

Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica
Sistemi operativi e reti
A.A. 2018-2019

Pietro Frasca

Lezione 19

Giovedì 13-12-2018

Algoritmi di sostituzione delle pagine

- Un algoritmo di sostituzione ottimo dovrebbe prevedere quali pagine saranno riferite nel futuro. Abbiamo già incontrato un problema analogo descrivendo l'algoritmo SJF nella schedulazione dei processi.
- Gli algoritmi reali, per predire il futuro, si basano sulle informazioni relative agli accessi effettuati nell'immediato passato.
- Infatti, a partire da un generico istante e per un certo intervallo di tempo, i processi generano indirizzi che sono spesso contenuti all'interno di un ristretto numero di pagine detto **working set** (insieme di lavoro).
- Il **working set** varia nel tempo, ma gradualmente.
- Le chiamate di funzioni provocano una variazione di **working set**.
- L'algoritmo di sostituzione più semplice è il **FIFO** che sceglie la pagina che risiede da più tempo in memoria. Tuttavia è poco efficiente.

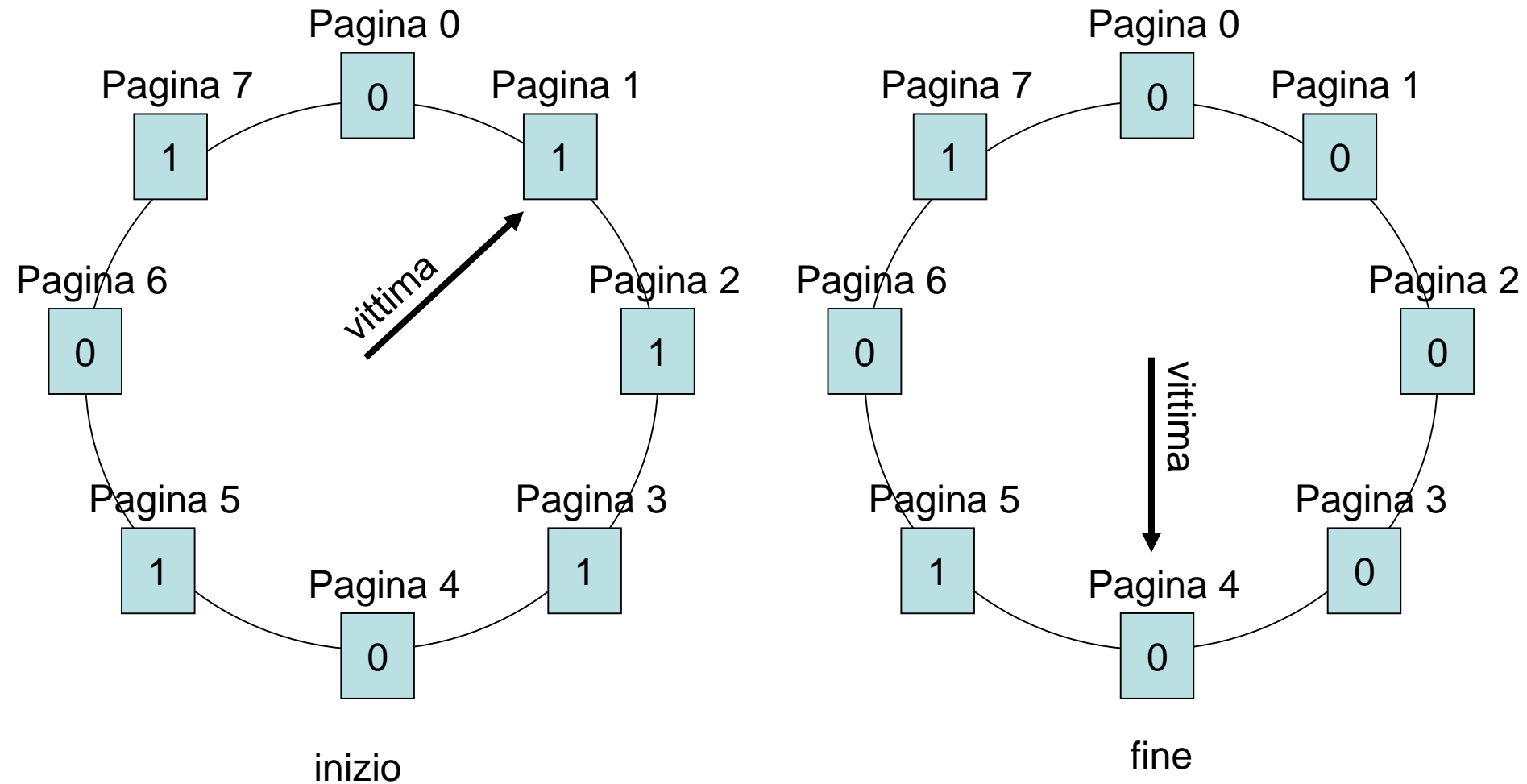
- Un algoritmo molto più complesso, ma molto più efficiente in termini di page-fault, è l'**LRU (Least recently Used)** che sceglie, per la sostituzione, la pagina **meno recentemente utilizzata** indipendentemente da quando è stata caricata in memoria.
- E' stato sperimentato che LRU è l'algoritmo che fornisce le migliori prestazioni. Tuttavia la sua realizzazione non è conveniente in quanto richiede un complesso supporto hardware e l'implementazione software produce un overhead troppo elevato.
- Molti algoritmi sono approssimazioni dell'LRU.
- Uno di questi è l'algoritmo **second-chance** (seconda scelta) chiamato anche **clock algorithm** (algoritmo dell'orologio).

Algoritmo seconda scelta

- Per consentire al gestore della memoria di fare statistiche sull'uso delle pagine si utilizzano i bit **U** (uso) e **M** (modifica) associati a ciascuna pagina. Questi due bit devono essere aggiornati ogni volta che una pagina è indirizzata; è quindi fondamentale che siano modificati via hardware.
- All'avvio di un processo, i due bit U e M di ogni pagina sono resettati (posti a 0).
- Periodicamente, ad esempio ogni 20 ms, all'interruzione di un timer, tutti i bit U sono resettati.
- Le pagine sono distinte in due classi: quelle con il bit **U=1** (le più recentemente usate) e quelle con il bit **U=0** (**le meno recentemente usate**). Si sceglie una pagina con politica FIFO, dapprima tra quelle con il bit U=0, se ci sono.

- La **tabella delle pagine fisiche** è gestita come un array circolare (round robin). Viene tenuta aggiornata una variabile di sistema **vittima**, contenente l'indice della pagina fisica successiva a quella che è stata sostituita per ultima.
- Quando si verifica un **page-fault** la verifica inizia con la pagina il cui indice è contenuto nella variabile **vittima**. Se tale pagina ha il bit **U=0** è scelta per la sostituzione, altrimenti il suo bit U è resettato e si prosegue fino a trovare una pagina che ha il bit U=0.
- Nella figura è mostrato il funzionamento dell'algoritmo disegnando la tabella delle pagine fisiche in forma circolare e mettendo in evidenza per ogni pagina solo il bit d'uso U.

tabella delle pagine fisiche.
Sono visualizzati **solo i bit d'uso U**



Algoritmo second-chance

- Per ridurre i trasferimenti tra memoria e disco, e quindi per migliorare le prestazioni dell'algoritmo, per la classificazione delle pagine si considera anche il bit di modifica **M**. In tal modo le classi diventano 4, relativamente alle combinazioni dei bit **U-M**:

classe0: 0-0	non usata – non modificata
classe1: 0-1	non usata- modificata
classe2: 1-0	usata – non modificata
classe3: 1-1	usata - modificata

- Rispetto alla scelta basata sul solo bit **U** si privilegiano le pagine con il bit di modifica **M=0** in quanto non modificate e quindi non è necessario riscriverle sul disco.
- La classe1, contiene pagine che sono state indirizzate, M=1 ma che hanno il bit U=0 per via del reset eseguito all'interruzione del clock.

- Sono molti gli algoritmi realizzati per la sostituzione delle pagine.
- Ogni tipo di algoritmo ha molte varianti in base ai criteri di scelta delle pagine da rimpiazzare. Ad esempio, al momento del page-fault, si potrebbe scegliere di revocare una pagina fisica dal processo che ha generato il page-fault (tecnica della **sostituzione locale**) oppure scegliere la pagina revocandola a qualsiasi processo (tecnica della **sostituzione globale**). L'algoritmo di sostituzione globale consente una scelta più conveniente, in quanto viene fatta su un numero maggiore di pagine.
- Molti sistemi, tra cui UNIX, non consentono di saturare completamente la memoria ma conservano un certo numero di pagine fisiche di riserva. In tal modo la gestione del page-fault avviene più velocemente. D'altra parte però il paginatore, quando verifica che la memoria sta per esaurirsi, deve salvare un numero di pagine in modo da renderle di nuovo libere.

- Per minimizzare il numero di page-fault, molti paginatori tengono traccia del working set di ciascun processo, in modo tale da caricarlo in memoria prima che il processo riprenda l'esecuzione. Questa tecnica è detta **working set model**.

Gestione della memoria nei sistemi Unix

- Lo spazio di indirizzamento di un processo Unix è segmentato, ed è costituito da tre segmenti separati: **codice**, **dati** e **stack**. Il segmento dati è diviso in due parti: i dati inizializzati e i dati non inizializzati (BSS, Block started by symbol). Tutte le variabili BSS sono inizializzate a zero dopo il caricamento.
- Il comando **size** visualizza le dimensioni dei segmenti codice (text), dati (data) e BSS di un file eseguibile .
- Molti programmi richiedono di allocare memoria dinamicamente durante l'esecuzione. La chiamata di sistema che consente l'allocazione dinamica della memoria è **brk**. La funzione di libreria C **malloc**, generalmente usata per allocare memoria, utilizza questa chiamata di sistema. L'area di memoria allocata in modo dinamico è detta **heap**.
- Il sistema di gestione della memoria utilizza il modello di segmentazione paginata. In particolare, l'allocazione dei segmenti avviene con la tecnica della paginazione su richiesta.

- Per ogni processo, la corrispondenza tra pagine virtuali e pagine fisiche è realizzata mediante la **tabella delle pagine**.
- Inoltre il kernel gestisce lo stato di allocazione delle pagine fisiche mediante la **tabella delle pagine fisiche** detta **core map**. Ogni elemento della **core map** specifica se la relativa pagina fisica è libera o allocata. Se la pagina fisica è allocata, l'elemento indica quale pagina virtuale è in essa contenuta e il PID del processo a cui appartiene.
- La sostituzione delle pagine è eseguito dal processo di sistema **page daemon** che utilizza una strategia che si basa sull'algoritmo di seconda scelta (orologio). Il page daemon ha PID pari a 2 e viene avviato da **init** che è il processo numero uno (PID=1).
- Il page daemon va in esecuzione periodicamente, ad esempio ogni 250 ms, ma interviene solo se il numero di pagine libere scende al disotto di una soglia prestabilita, il cui valore è stabilito dalla variabile di sistema **lotsfree**.

- Se l'intervento del page daemon non è sufficiente a mantenere bassa la frequenza di paginazione, nonostante il numero di pagine libere rimanga sotto la soglia stabilita da `lotsfree`, interviene lo **swapper** che sposta un certo numero di pagine di processi dalla memoria all'area di swap su disco.

Gestione della memoria secondaria

- Dispositivi di grande importanza
- forniscono il supporto per la memoria virtuale
- supporto alla memorizzazione dei file
- L'efficienza e l'affidabilità dei dischi si riflette sull'intero sistema
- Probabilmente nell'immediato futuro gli SSD sostituiranno completamente gli HD i quali tuttavia sono ancora molto diffusi

Organizzazione fisica dei dischi

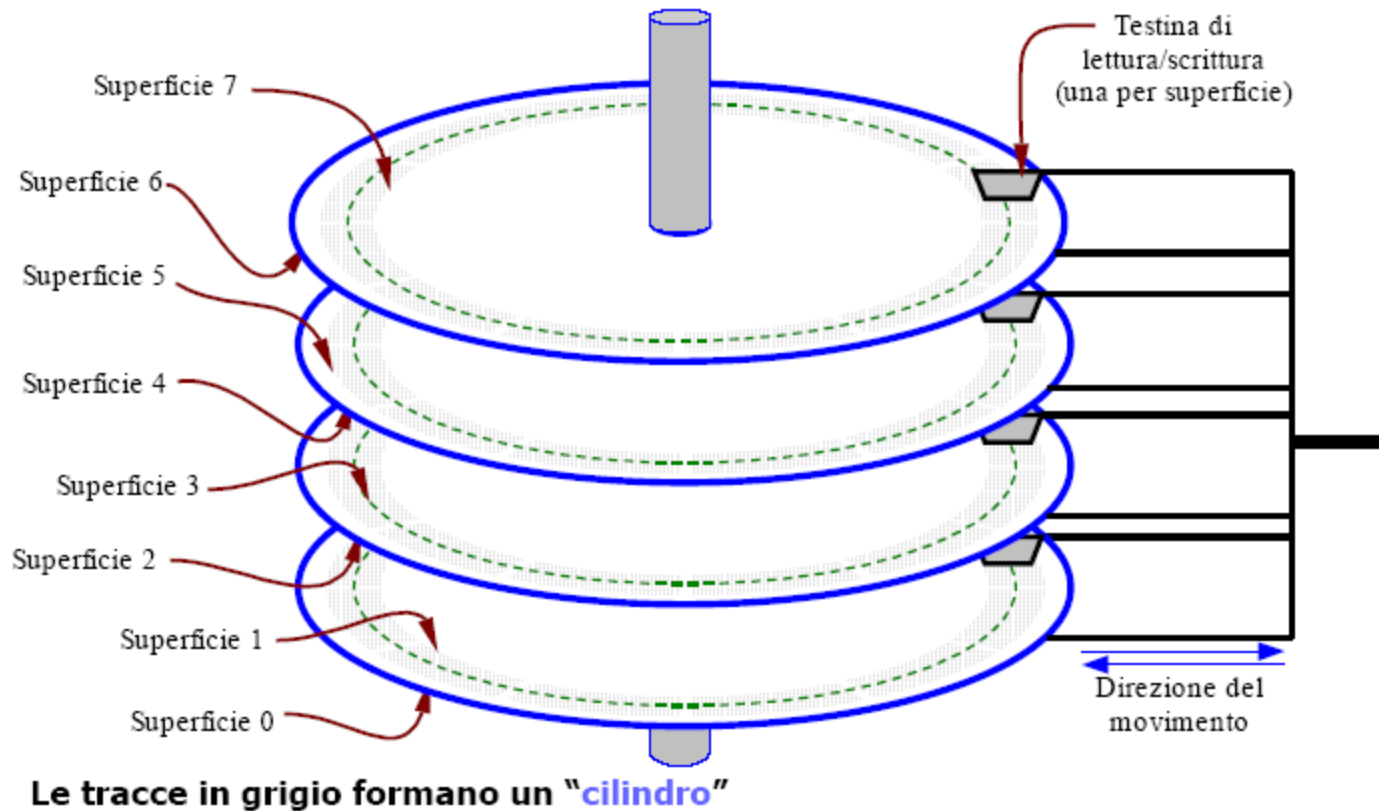
- Un disco è formato da vari dischetti tipicamente di alluminio o plastica, ricoperti di materiale magnetico.
- le testine possono essere fisse (una per ogni traccia) o mobili (una per ogni faccia del disco)
- Le testine mobili si muovono radialmente
- Sia due tracce adiacenti che due settori adiacenti sono separati da un piccolo spazio, privo di materiale magnetico (intertrack gap e intersector gap)

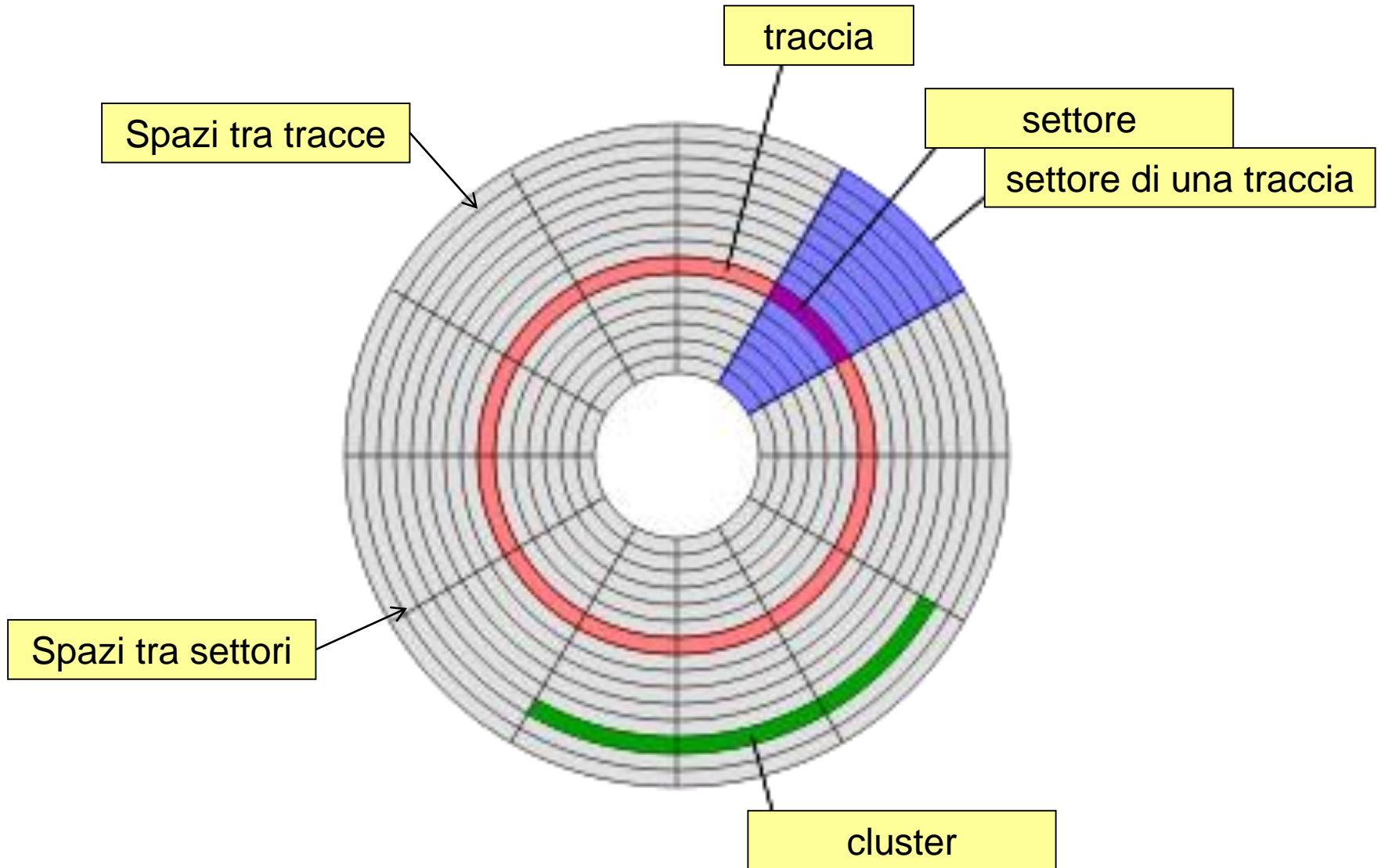
- Generalmente, il numero di dati memorizzabili su ogni traccia è costante, quindi la densità è maggiore per le tracce più interne. Questa caratteristica consente di semplificare la logica di controllo del disco.
- Interfacce standard di dischi molto diffuse sono ATA (Advanced Technology Attachment) conosciuta anche come IDE, che può essere Parallel ATA (PATA, meno recente) o Serial ATA (SATA, più recente) e SCSI (small computer system interface).
- Le dimensioni fisiche del disco fisso sono 3,5" o 2,5". Nei PC desktop i dischi sono praticamente tutti in formato 3,5", con l'attacco PATA sempre meno utilizzato a favore del più veloce SATA.
- La maggior parte dei dischi fissi per desktop hanno una velocità di 7.200 rpm (120 rps), ma esistono modelli particolarmente veloci da 10.000 rpm.

- Le capacità attualmente più diffuse sono comprese tra i 500 GB a qualche TB.
- Nei [notebook](#), si usano prevalentemente dischi da 2,5", che nella maggior parte dei casi raggiungono la velocità massima di 5.400 rpm, con pochissimi modelli da 7.200 rpm. Anche le capacità sono inferiori, e sono comprese tra 160 e 1 TB.
- La grande novità nel settore è costituita dai Solid State Drive (SSD), che man mano sostituiranno gli HD magnetici.
- I principali vantaggi di queste unità sono: minor peso e dimensioni, nessuna rumorosità, consumi e tempi di accesso ridotti. La velocità di trasferimento è intorno ai 500 MB/sec.
- Unico limite, al momento, il costo per GB di molto superiore ai classici hard disk.



Interno di un disco rigido





- il trasferimento di dati tra disco e memoria avviene a gruppi di dimensione multipla di quella del settore (cluster).
 - Il disco prima dell'uso deve essere formattato: sono memorizzati nei settori alcuni dati di controllo che consentono al controllore di identificare le tracce e i settori.
 - Tipicamente i dischi hanno due facce per ciascun piatto.
 - Il **cilindro** è l'insieme di tutte le tracce concentriche che formano l'HD.
 - Il blocco è l'insieme di tutti i settori che occupano la stessa posizione nelle diverse tracce.
- Un settore di una traccia costituisce l'unità minima di memorizzazione dei dati. Il suo indirizzo **ls** è una funzione dei tre parametri: **f**, (faccia), **t** (traccia o cilindro) e **s** (settore):

$$ls = f(f,t,s);$$

$$ls = NT*f + NS*t + s$$

NT: numero di tracce per faccia (o testina)

NS: numero di settori per traccia

- In base a tale funzione un disco è visto logicamente come un array di blocchi contigui:

Faccia 0 Traccia 0 settore 0	settore 0
Faccia 0 traccia 0 settore 1	settore 1
Faccia 0 traccia 0 settore 319	settore 319
Faccia 0 traccia 10 settore 1	settore 3201
Faccia 7 traccia 13613 settore 319	

Parametri caratteristici di un disco

Numero di cilindri	13614
Tracce per cilindro	8
Settori per traccia	320
Byte per settore	512
Capacità	18.3 GB
Tempo minimo di seek	0.6 ms
Tempo medio di seek	5.2 ms
Tempo di rotazione	6 ms
Tempo di trasf. di un settore	19 us

Criteri di ordinamento dei dati su disco e politiche di scheduling

- Per valutare le prestazioni di un disco si ricorre spesso al **tempo medio di trasferimento (Tmt)**, che indica il tempo medio necessario per effettuare la scrittura o la lettura di una certa quantità di byte.
- Il **Tmt** dipende da due parametri:
 - 1) il **tempo medio di accesso (Tma)** costituito dal tempo che la testina impiega per posizionarsi in corrispondenza del settore desiderato, e
 - 2) il **tempo di trasferimento (Tts)** vero e proprio necessario per trasferire i dati del settore.

$$Tmt = Tma + Tts$$

- Il tempo medio di accesso **Tma** a sua volta, dipende da due fattori
 - 1) **tempo medio di seek (Tseek)** che è il tempo necessario per spostare la testina in corrispondenza della traccia contenente il settore desiderato;
 - 2) **Latenza di rotazione (Rotational Latency, Trl)** che è il tempo di rotazione che il disco impiega per trovarsi sul settore.

$$Tma = Tseek + Trl$$

La relazione precedente diventa:

$$Tmt = Tseek + Trl + Tts$$

- Il tempo **Tts** può essere approssimato, trascurando gli spazi tra settori (intersector gap), al valore **Trot/ns** dove **ns** indica il numero di settori per traccia e **Trot** è il tempo di rotazione che indica il tempo necessario per compiere un giro del disco.

- Per il disco dell'esempio si ha che:

$T_{ts} = 6 \text{ ms} / 320 = 0,01875 \text{ ms}$ che è circa $19 \mu\text{s}$

- E' evidente che il tempo medio di trasferimento dipende fondamentalmente dal tempo medio di accesso **Tma** e quindi da **Tseek** e **Trl**.
- Per ridurre il **Tmt** è necessario agire su due aspetti:
 - Criteri di memorizzazione dei dati su disco;
 - Politiche di scheduling per l'accesso al disco da parte dei vari processi.