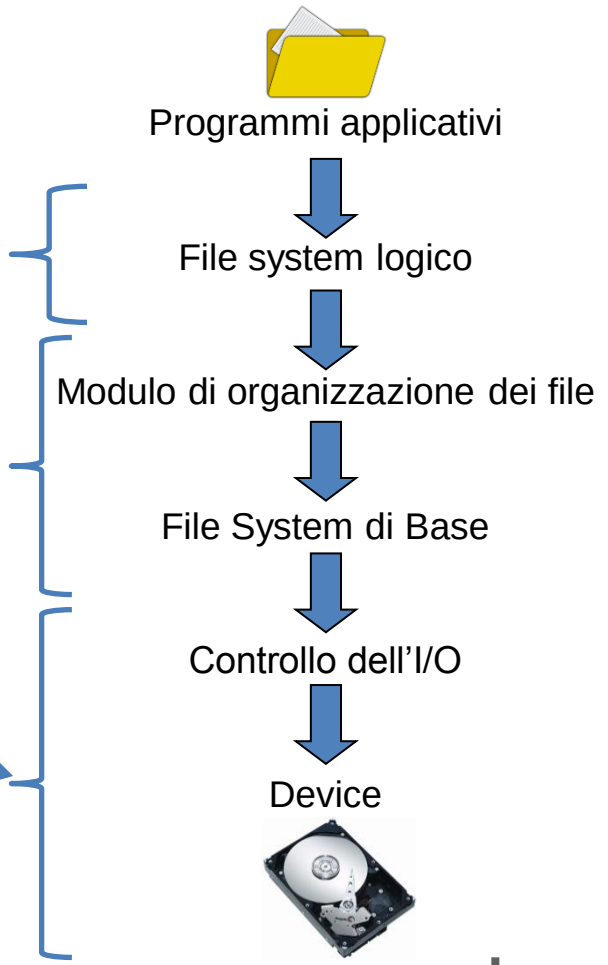
Memoria di Massa
(memoria secondaria)

Sommario

- Struttura fisica della memoria secondaria
- Formattazione
- Blocco di avvio
- Gestione dell’unità disco (scheduling)
- Blocchi difettosi

File system e memoria di massa

- Dal punto di vista logico il file system si compone di tre parti:
  - L'interfaccia al programmatore
  - Le strutture dati interne e gli algoritmi di allocazione
  - Il livello più basso legato alla memorizzazione su memoria secondaria

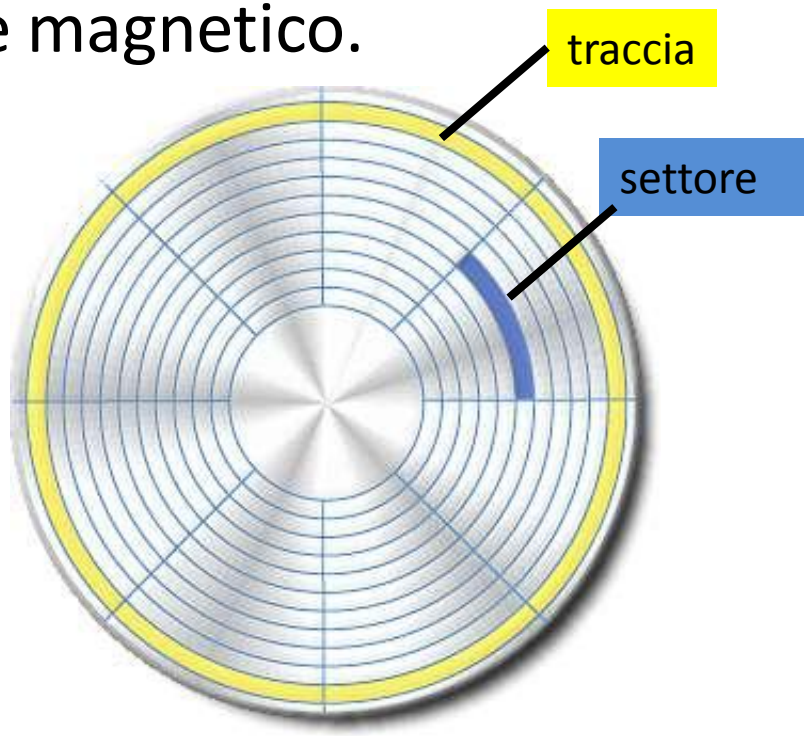


Memoria secondaria

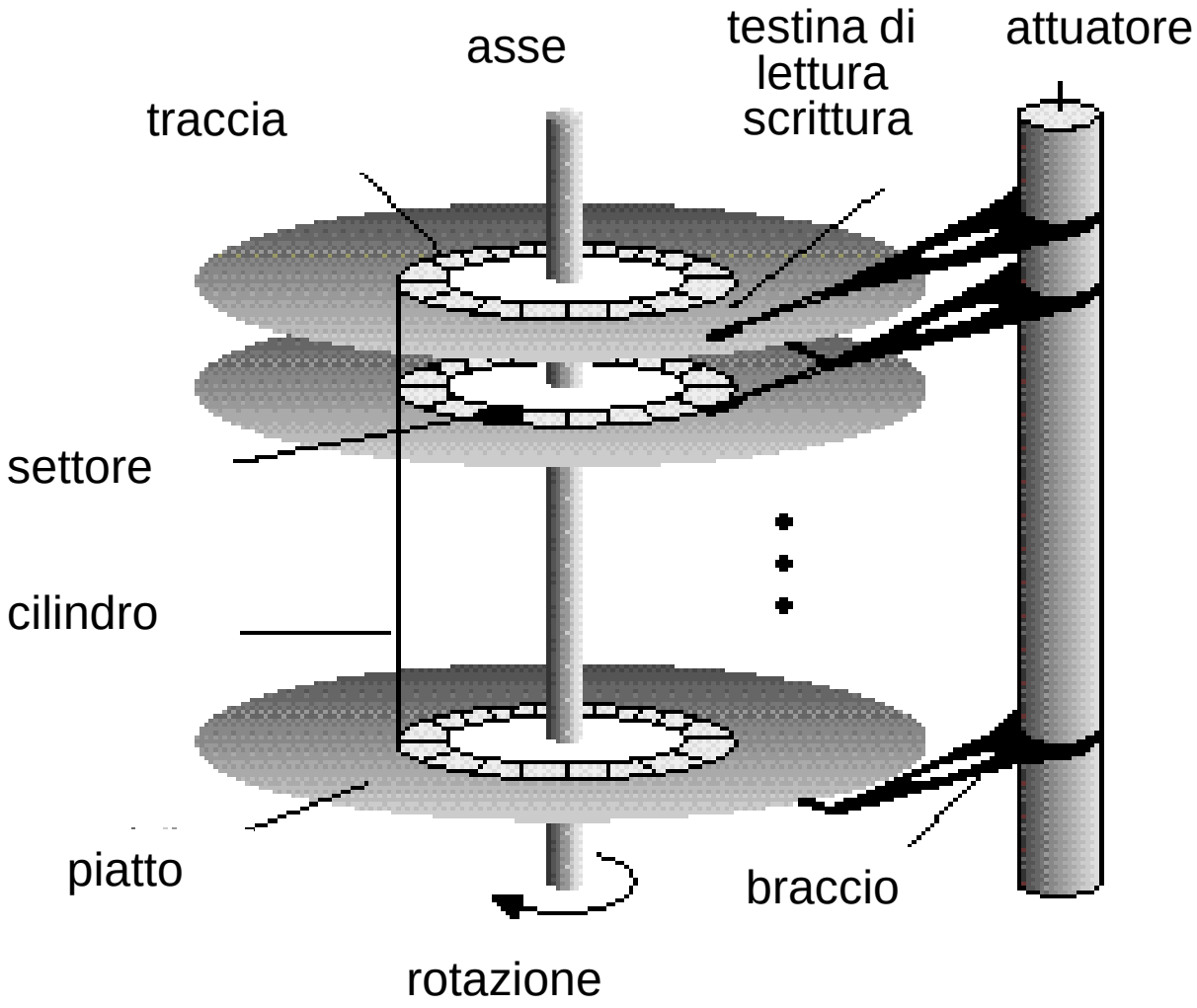
- Il file system si dispiega fornendo una interfaccia logica comune a dispositivi di memoria di massa che hanno caratteristiche fisiche diverse.
- In particolare vedremo due diversi tipi di dispositivi di memoria di massa che hanno strutture fisiche molto differenti
  - Dischi fissi veri e propri, ovvero dischi magnetici
  - Dischi a stato solido, o meglio dispositivi a stato solido (SSD, Solid State Drive)

Struttura dei dischi

- Dal punto di vista fisico, i dischi magnetici sono composti da un insieme di **piatti**, superfici circolari con un diametro che solitamente va da 1,8 a 5,25 pollici e ricoperte da materiale magnetico.
- La superficie è divisa in **tracce** circolari che a loro volta sono divise in **settori**.
- L'insieme delle tracce equidistanti dal centro del disco, costituisce un cilindro.



Struttura dei dischi



Dischi magnetici

Struttura dei dischi

- Dal punto di vista dell'indirizzamento il disco può essere considerato un **vettore monodimensionale** di **blocchi logici**, in cui il blocco logico è l'unità minima di trasferimento.
- Il vettore monodimensionale di blocchi logici è mappato sequenzialmente sui **settori** del disco:
  - Dalla traccia 0 del cilindro più esterno
  - Lungo la traccia
  - Verso i cilindri più interni

Lettura/scrittura

- Le testine di lettura/scrittura sono sospese sopra ogni superficie di ogni piatto e sono attaccate ad un braccio che le muove **in solido**, cioè tutte insieme.
  - scrittura: passaggio di corrente positiva o negativa attraverso la testina magnetizza la superficie
  - lettura: passaggio sopra un'area magnetizzata induce una corrente positiva o negativa nella testina.
- Il movimento meccanico è alla base del funzionamento del disco magnetico…

Settore

- Il **settore** rappresenta l'unità più piccola di informazione che può essere letta e scritta sul disco:
  - Solitamente ha una dimensione di 512 byte (ma varia da 32B a 4KB)
  - Per **accedere a un settore** bisogna specificare **la superficie, la traccia e il settore**
- I settori possono essere raggruppati dal punto di vista logico in **cluster** per motivi di efficienza:
  - la dimensione del cluster dipende dal SO e dal disco, va da 2KB a 32KB
  - Un file occupa sempre almeno un cluster

Struttura dei dischi

- Non è banale partire dall'indirizzo del blocco logico e calcolare il settore fisico corrispondente perché la sequenza di mappaggio dei blocchi logici in settori dipende da vari fattori:
  - Possono esserci **settori danneggiati**.
  - Le tracce vanno via via restringendosi, offrendo spazio a un numero via via calante di settori.
- Nei dischi le tracce sono caratterizzate per appartenenza a diverse aree del disco, ovvero ad aree diverse corrispondono **tracce con un diverso numero di settori**. Le tracce nelle aree più esterne del disco possono contenere fino al 40% in più dei settori, rispetto alle aree più interne al disco

Gestione del memoria secondaria

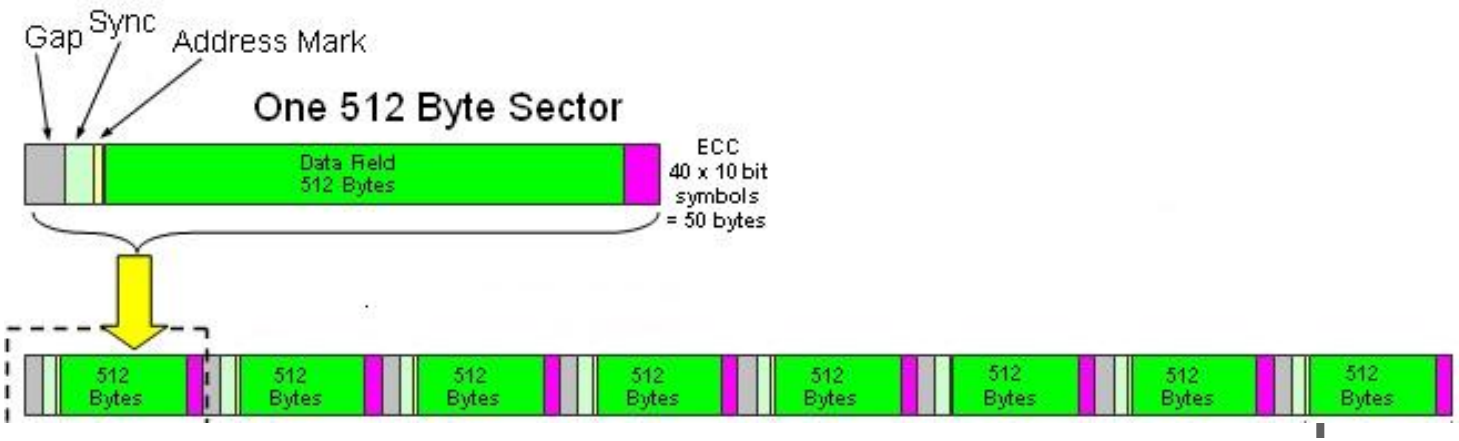
- Il SO si deve occupare anche di:
  - **Inizializzare i dispositivi di memoria secondaria**, mediante **formattazione** sia di tipo fisico (formattazione di basso livello) che di tipo logico.
  - Creare il **blocco di boot**, che si occupa dell'avvio della macchina e del SO.
  - Gestire i **blocchi difettosi** del disco.
  - Gestire in modo efficiente l'**area di swap**.

Formattazione

- Alla produzione il disco magnetico non è diviso in settori e dunque non può essere letto o scritto dal controller.
- L'operazione con cui il disco viene diviso in settori (e con cui a ciascun settore è spezzato in intestazione, area dati e cosa) è detta **formattazione fisica** o **formattazione di basso livello.**

Formattazione

- L'area dati è tipicamente di 512 byte. Intestazione e coda servono al controller per inserire informazioni di servizio
- Per esempio il codice di correzione degli errori (**ECC**, **Error-Correcting Code**) che è usato dal controller per rilevare malfunzionamenti sul settore. Se il codice calcolato run-time dal controller è diverso da quello memorizzato, allora si è verificato un errore su quel settore.



Formattazione

- Il disco formattato a basso livello è utilizzabile dalla macchina ma non dall'utente.
- Per renderlo visibile all'utente il SO deve inserire le proprie strutture dati, creando le partizioni e le corrispondenti directory in aggiunta alle strutture per la gestione dello spazio non allocato.
- Questa operazione è detta **formattazione logica**.

Formattazione

- Riassumendo:
  - Disco vergine

  - Formattazione fisica:

    - Suddivisione in settori
    - Identificazione dei settori
    - Aggiunta di spazio per correzione di errori (ECC)
  - Formattazione logica:

    - File system
    - Lista spazio occupato (FAT) e libero (Free List – FL)

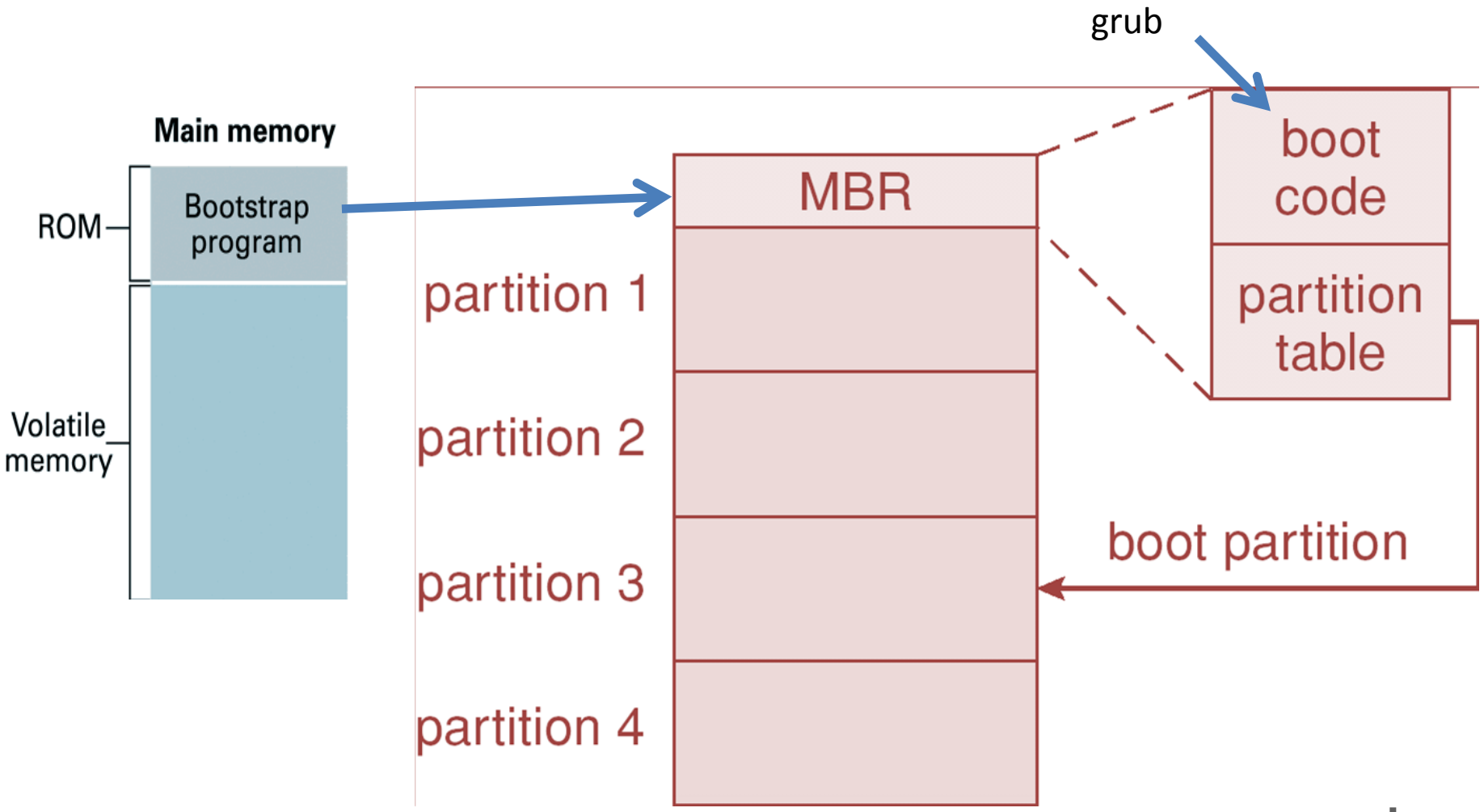
Blocco di boot

- Al momento dell'accensione (o del riavvio) il computer mette in funzione un programma iniziale (**bootstrap**) molto semplice che inizializza il sistema e avvia il SO.
- Il bootstrap può essere memorizzato:
  - Su una ROM (sola lettura) ma è difficile sostituire il codice per migliorarlo.
  - Su disco: in questo caso viene utilizzata un'area del disco detta **area di boot**.

Blocco di boot

- In realtà l'approccio è misto:
  - Nella ROM è collocato un piccolissimo programma che avvia il boot (**bootstrap loader**).
  - Il programma di avvio completo è **nell'area di boot** del disco.
- Ogni disco che contiene un'area di boot è detto **disco di sistema** o **unità di avvio**.
- Nota bene: quando viene letta l'area di boot il SO non e' ancora stato caricato, non ha i driver e **non esiste il file system.** Nel settore di boot quindi non esiste una strutturazione logica a file. Quindi l'accesso all'area di boot non è una normale lettura di un file bensi' la lettura di alcuni settori fisici in cui e' contenuta direttamente l'immagine da caricare in memoria.

Boot


grub
Main memory
ROM
Bootstrap program
Volatile memory
MBR
partition 1
partition 2
partition 3
partition 4
boot code
partition table
boot partition

Avviamento

- Il primo settore del disco fisso, chiamato Master Boot Record (MBR) contiene il codice di avviamento e la tabella delle partizioni.
- Una o più delle partizioni specificate dalla tabella possono essere partizioni d'avviamento e contenere il SO e i driver dei dispositivi.

Dischi magnetici

- Un disco può essere rimuovibile, ovvero può essere collegato e scollegato dal sistema (o inserito/rimosso) in base alle necessità.
- L'unità disco è connessa al sistema attraverso un bus di I/O:
  - IDE-ATA
  - SATA (Serial ATA)
  - FC (Fiber Channel)
  - SCSI
  - USB (plug and play)

Prestazioni dei dischi magnetici

- Quando il disco è in funzione, ruota ad alta velocità (max circa 15.000 giri al minuto)
- Le prestazioni del disco sono in realtà caratterizzate da due fattori:
  - La **velocità di trasferimento**: ovvero i dati con cui i dati vanno da disco a memoria centrale e viceversa tramite dei bus.
  - La **velocità di posizionamento**, ovvero il tempo che la testina impiega a posizionrasi sul settore giusto. È composto da:
    - **Tempo di seek**: che serve al braccio per muoversi sulla traccia giusta.
    - **Latenza di rotazione**: che serve al disco per ruotare finchè la testina non è sul settore giusto.

Scheduling del disco

- Considerato che il sistema deve servire molti processi contemporaneamente, occorre agire sull'ordine con cui questo servizio viene fornito in modo da minimizzare il tempo di seek complessivo, cioe' massimizzare la velocita' di posizionamento, poiche' non possiamo fare nulla per quanto riguarda la velocita' di trasferimento mediante i bus.
- Dobbiamo cioè implementare una politica di **scheduling del disco**.

Scheduling del disco

- Ogni singola richiesta di lettura scrittura avviene attraverso una syscall bloccante di I/O che specifica tra l'altro:
  - Se l'operazione è di **input** o di **output**.
  - L'**indirizzo del disco** su cui effettuare il trasferimento.
  - L'**indirizzo di memoria** su cui effettuare il trasferimento.
  - Il **numero di byte** da trasferire.

Scheduling del disco

- Se il disco è già impegnato a soddisfare richieste precedenti, ogni nuova richiesta viene accodata a quelle sospese.
- E' quindi possibile su questa coda realizzare una **politica di scheduling del disco** scegliendo quale delle richieste pendenti debba essere servita per prima.
- Scopo della politica di scheduling deve essere l'efficienza del sistema:
  - Rendendo minimo il tempo di accesso complessivo, ovvero
  - Rendendo minimo il tempo di seek complessivo.

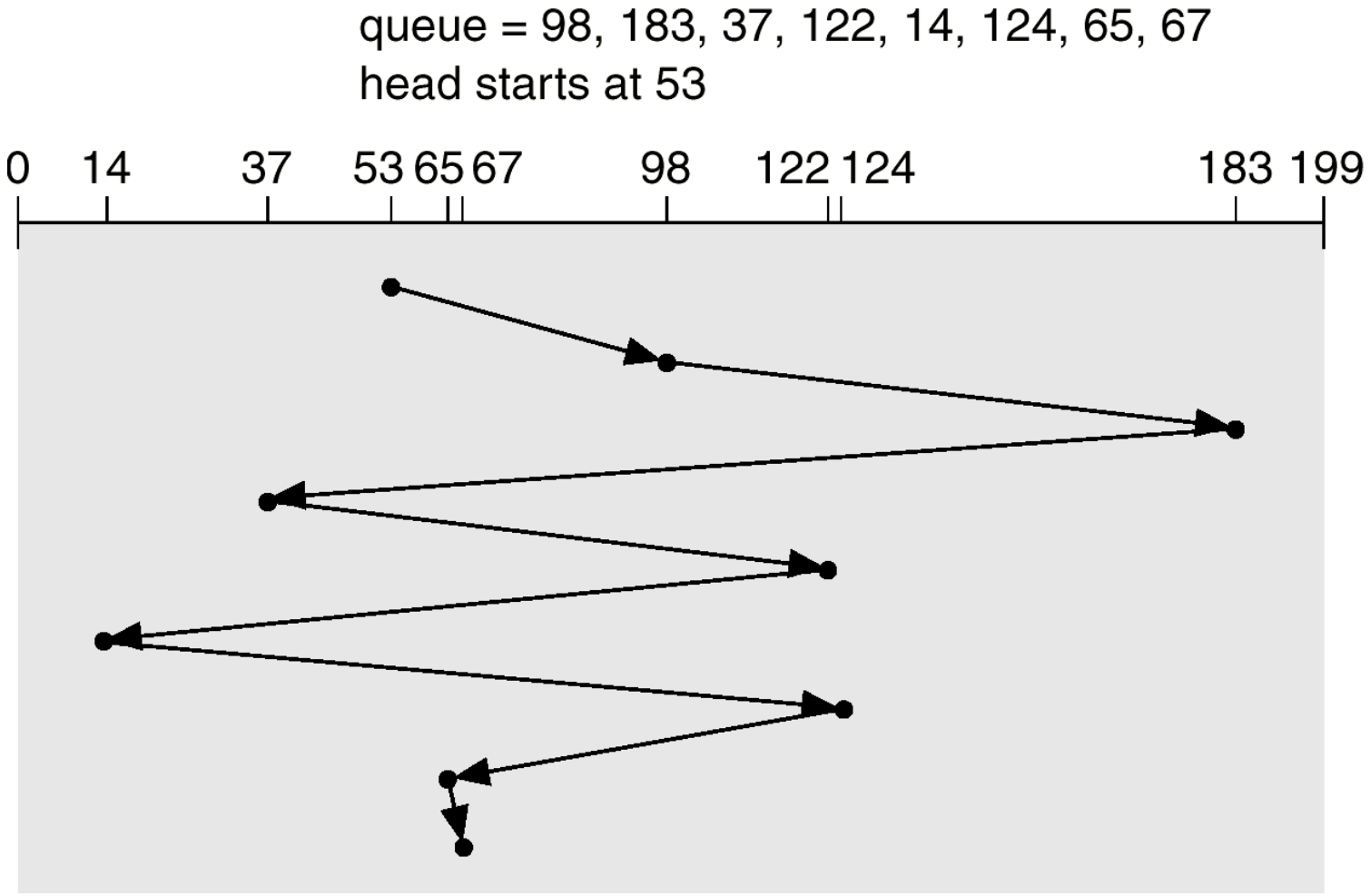
Scheduling del disco

- Come abbiamo fatto per le politiche di virtualizzazione della memoria, anche in questo caso verificheremo le prestazioni di diversi algoritmi sulla base di una situazione standard
- La descriviamo indicando:
  - **Elenco delle richieste pendenti** (in ordine di arrivo, indicate con il numero corrispondente di cilindro):
    98, 183, 37, 122, 14, 124, 65, 67
  - **Posizione della testina**: cilindro 53
  - Per semplicita' non consideriamo il settore da leggere.

FCFS

- La forma più semplice di scheduling è rappresentata dall'algoritmo **First Come, First Served** (**FCFS**), in cui la prima richiesta arrivata è anche la prima ad essere servita.
- Si tratta di un algoritmo intrinsecamente equo (fair), che non risulta però particolarmente efficiente perché non tiene conto della struttura del disco.
- Sulla sequenza di prova si sposta su 640 cilindri.

FCFS


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

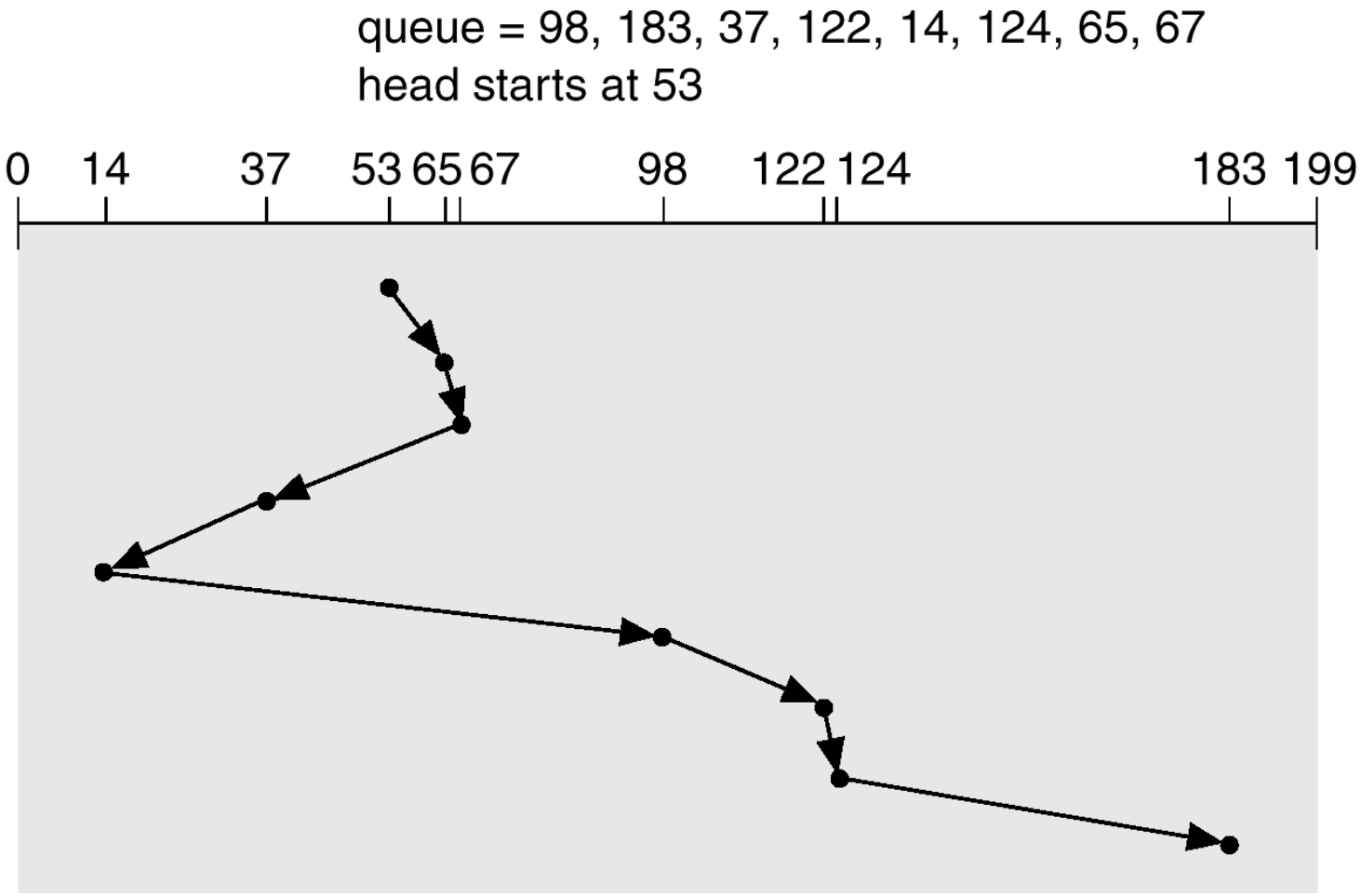
SSTF

- La scarsa efficienza di FCFC è dovuta al fatto che non viene considerata la prossimità delle richieste alla posizione corrente della testina.
- Sembra ragionevole poter migliorare notevolmente le prestazioni servendo prima tutte le richieste che sono vicine alla posizione attuale della testina.
- Su questa ipotesi è basato l'algoritmo **Shortest Seek Time First** (SSTF).

SSTF

- L’algoritmo procede scegliendo ad ogni istante la richiesta che necessita del tempo di ricerca più breve ovvero quella che riferisce al cilindro più vicino, rispetto alla posizione attuale della testina.
- Il miglioramento ottenuto rispetto a FCFS è notevole. Per esempio sulla sequenza di prova la testina si è spostata di 236 cilindri (circa un terzo di FCFC).

SSTF


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

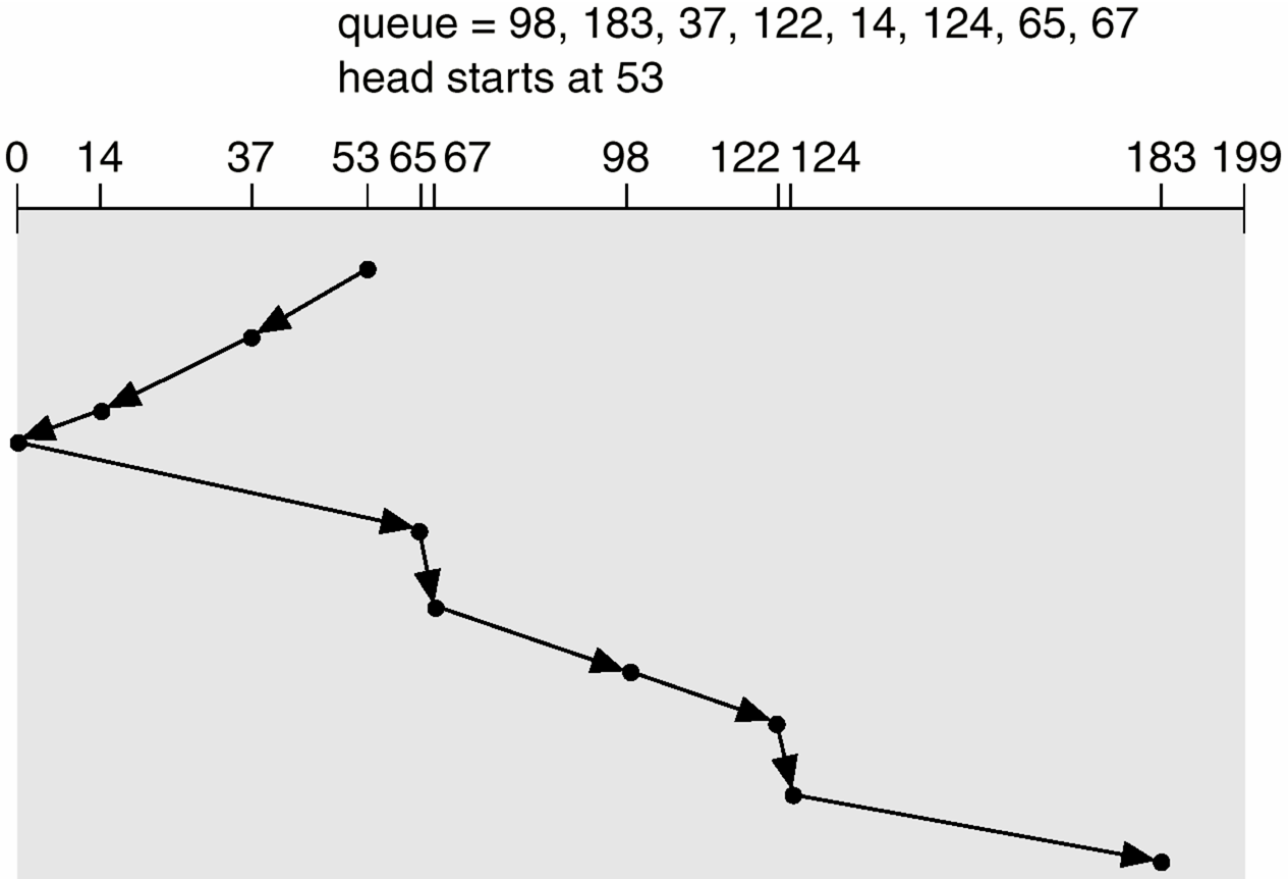
SSTF

- L'algoritmo soffre però di **starvation**: se si susseguono richieste per una certa area del disco, la testina rimane sempre in quell'area e non vengono mai soddisfatte le richieste periferiche.
- La starvation è più frequente se il disco è molto caricato.
- L'algoritmo inoltre provoca frequenti inversioni di direzione che risultano poco efficaci in alcune situazioni (per esempio nel caso del movimento da 53-67-37).

SCAN

- L'algoritmo nominato **SCAN** (**scansione**) supera i due limiti individuati di SSTF servendo le richieste lungo una direzione di attraversamento del disco.
- Il braccio è spostato da un estremo all'altro del disco servendo mano a mano le richieste che vengono incontrate. La direzione viene invertita solo a fine corsa.

SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN

- L'algoritmo di SCAN è anche chiamato **algoritmo dell'ascensore** perché emula il comportamento di un ascensore con prenotazione che serva prima tutte le richieste in discesa e poi tutte quelle in salita.
- L'algoritmo è moderatamente fair:
  - Se una richiesta avviene immediatamente dopo il passaggio della testina, è sfavorita e rischia di attendere molto.
  - L'attesa è comunque finita perché l'algoritmo prima o poi incontra la fine del disco e inverte la direzione.

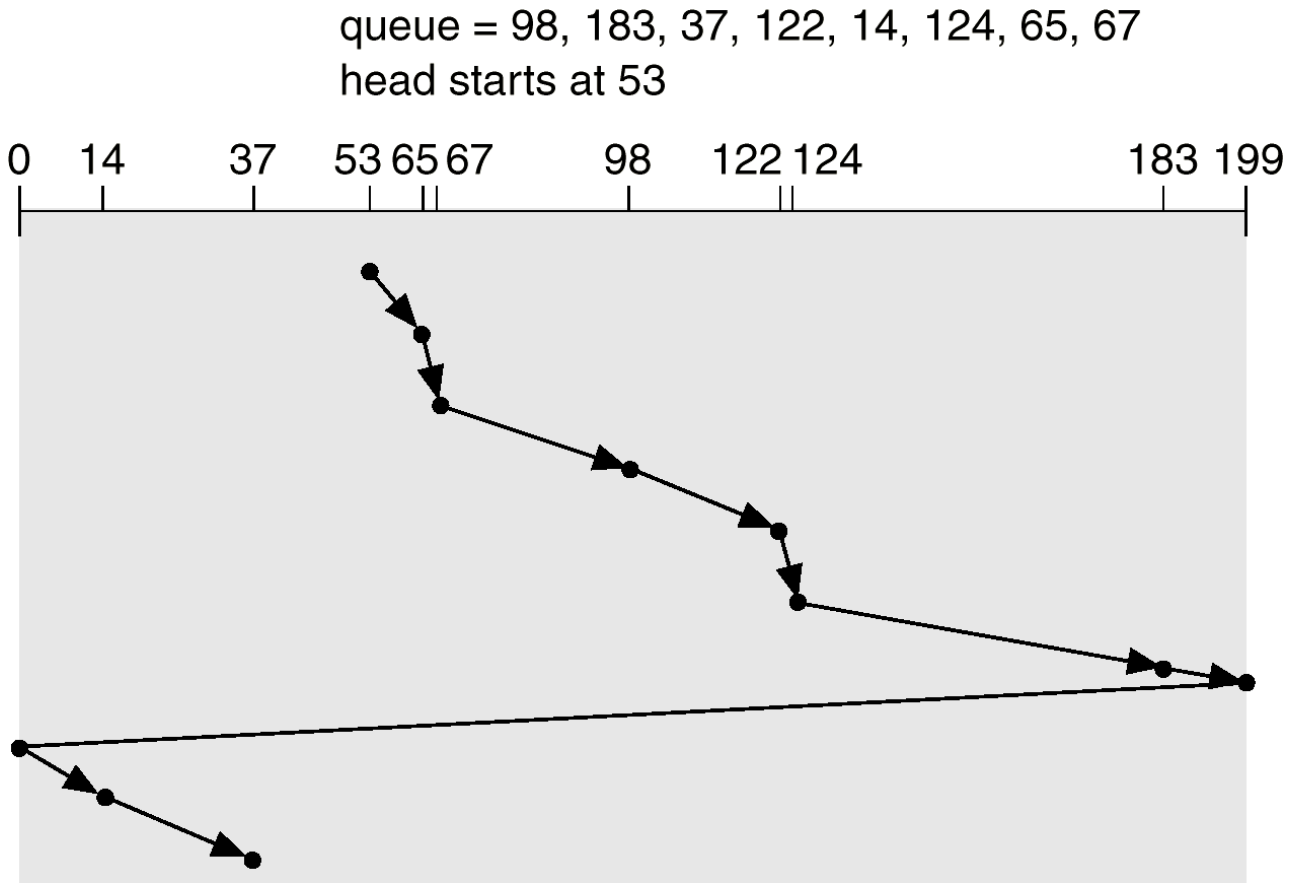
SCAN

- Inoltre pare ragionevole supporre che le richieste siano distribuite uniformemente lungo tutto il disco.
- Questo significa che quando la testina inverte la direzione e ripassa su tracce da pochissimo visitate probabilmente non troverà nessuna richiesta.
- Sulla base di questa osservazione è stato progettato l'algoritmo **C-SCAN** (SCAN Circolare).

C-SCAN

- **C-SCAN** opera come SCAN ma non inverte mai la direzione per cui quando la testina arriva a fine corsa, si riposiziona sulla traccia zero senza servire le richieste durante il viaggio di ritorno.
- Il disco è quindi trattato come una lista circolare considerando il primo e l'ultimo elemento come adiacenti.

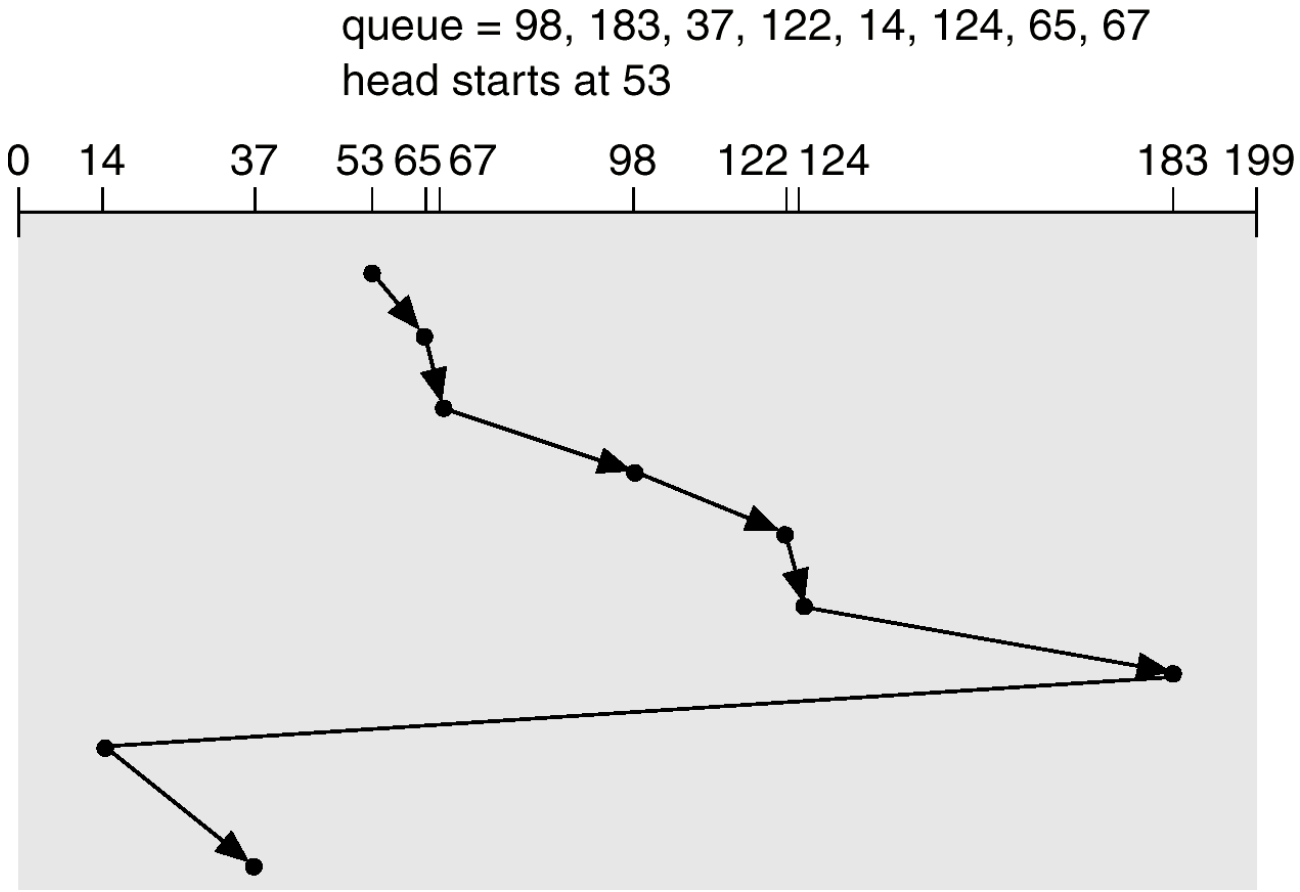
C-SCAN


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

- Un ulteriore miglioramento consiste nel fermare la testina quando viene incontrata l'ultima richiesta ed iniziare da quel punto il riposizionamento, risparmiando il tempo (inutile) necessario per raggiungere l'ultima traccia del disco.
- L'algoritmo che implementa questa politica è una variazione di C-SCAN, detta C-LOOK.

C-LOOK


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Quale algoritmo scegliere

- Il più efficiente è SSTF se il disco non è molto utilizzato, altrimenti tipicamente si utilizza C-LOOK.
- Occorrerebbe inoltre considerare:
  - Il metodo scelto per l'allocazione dei file (contiguo piuttosto che concatenato o indicizzato)
  - La posizione in cui sono memorizzate le directory, che sarà acceduta molte volte

Altre considerazioni

- Tutte le valutazioni fatte hanno riguardato i tempi di ricerca e non hanno considerato i tempi di latenza dovuti alla rotazione (cioe' non abbiamo considerato i settori).
- Questo tipo di valutazione è più complessa perchè l'hardware non mette a disposizione in modo trasparente la posizione fisica dei blocchi.
- Per risolvere questo problema esistono implementazioni hardware dello scheduling del disco.

Implementazioni HW

- PRO:
  - Il SO delega lo scheduling del disco al controller e dunque scarica la CPU (e il SO stesso) da un compito gravoso.
  - Il controller può usare algoritmi che ottimizzano sia in termini di tempo di ricerca che in termini di tempo di rotazione.
- CONTRO:
  - Il SO non può intervenire nella gestione della priorità delle richieste. Ad esempio una richiesta di swap di una pagina deve avere maggiore priorità rispetto alla scrittura di un file.
  - Il SO non riesce a mantenere la gestione dei blocchi fisici e quella dei file in modo coerente.

Le unità a stato solido

- Le unità a stato solido (**SSD**, **Solid State Drive**) non sono dischi, il termine disco a stato solido è improprio: la SSD non contiene nessun disco, ne magnetico né ottico.
- Gli SSD si basano su flash memory e non richiedono componenti meccaniche e magnetiche (aumenta quindi notevolmente l'affidabilità).
- Si tratta quindi di un dispositivo di memoria secondaria (alternativo ai dischi) che ha una struttura completamente differente