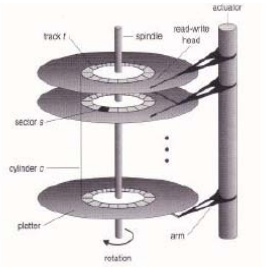
SISTEMI OPERATIVI

08.c

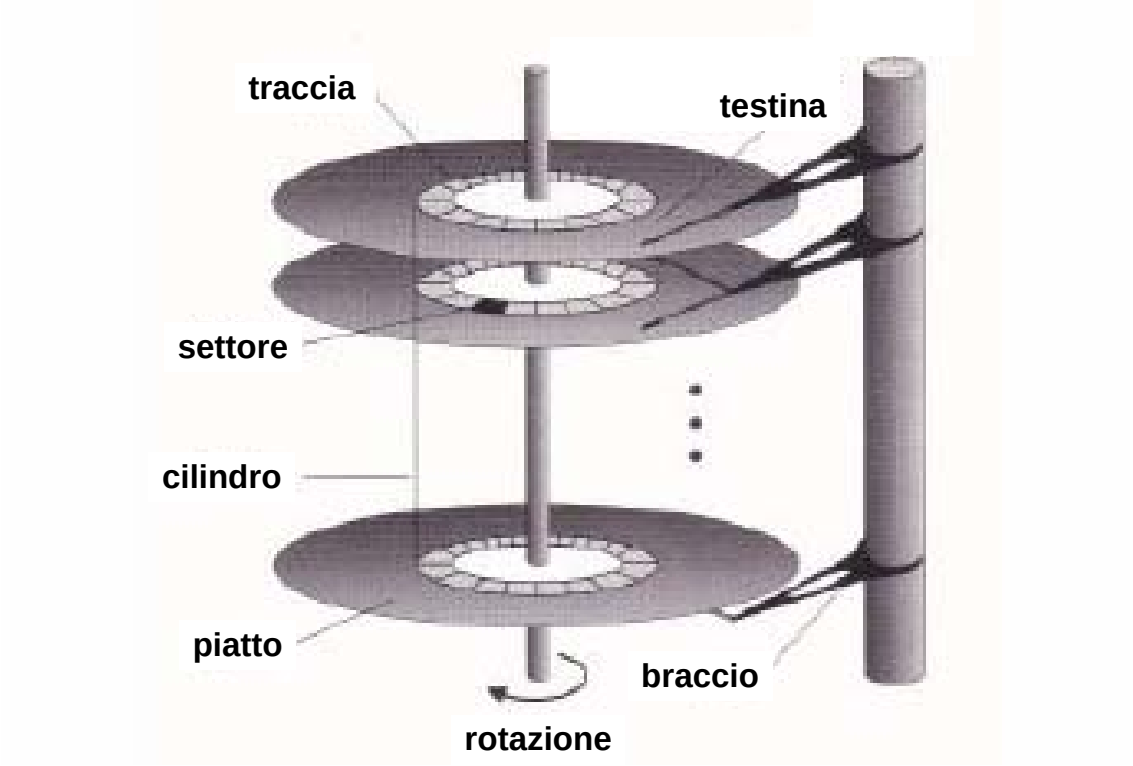
Gestione dei dischi e sistemi RAID

Gestione dei dischi

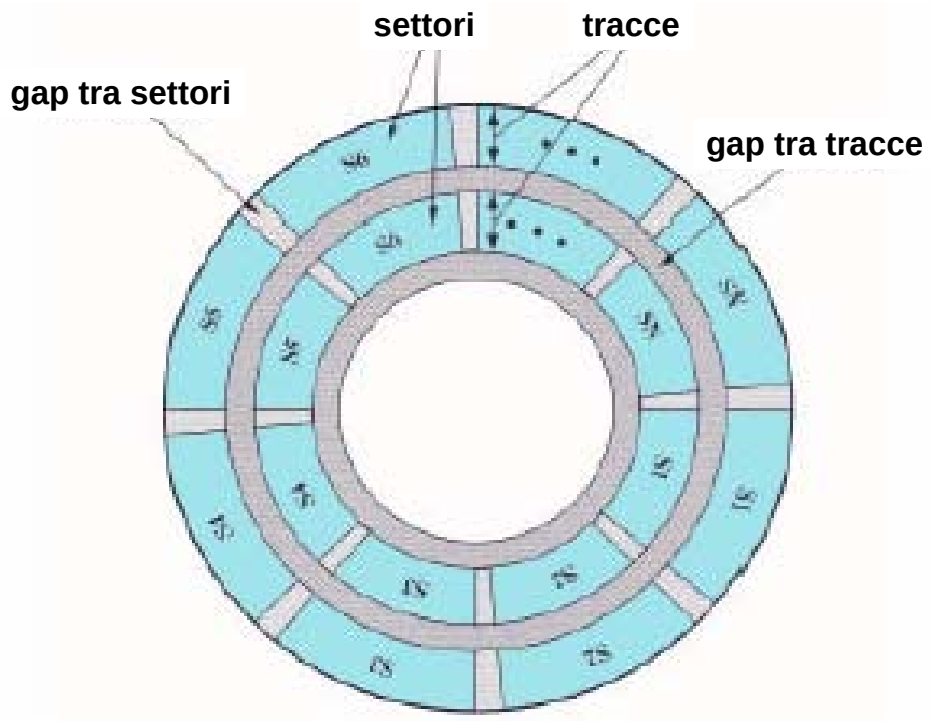
- Caratteristiche dei dischi magnetici
- Schedulazione degli accessi al disco
- Sistemi RAID



Struttura meccanica



Settori e tracce in una superficie



Caratteristiche

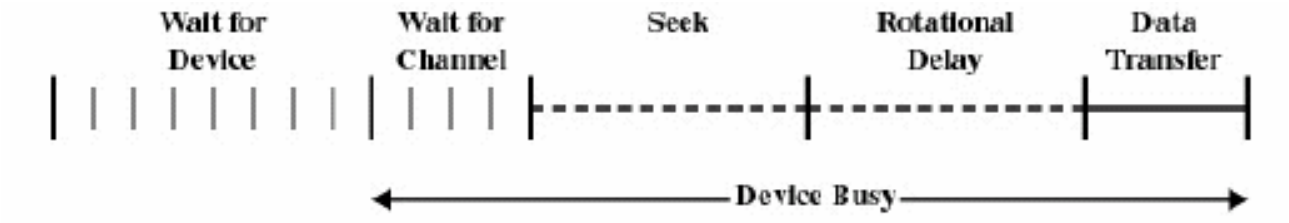
- I dischi sono molto più lenti della memoria centrale:
  - tempo di accesso tipico per i dischi: 10ms
  - tempo di accesso tipico per la memoria: 10ns
- Gli accessi al disco sono spesso molto frequenti :
  - richieste dei processi
  - richieste del SO (swap di pagine della memoria virtuale)
- Le prestazioni complessive del sistema dipendono fortemente dall'efficienza dell'I/O su disco.

Parametri e prestazioni

- Prima di effettuare un'operazione di lettura o scrittura su un settore è necessario posizionare la testina di lettura/scrittura sulla traccia e all'inizio del settore:
  - tempo di seek: per posizionare la testina sulla traccia:
    - t_S = (# tracce)xc + tempi di accelerazione/decelerazione (c=tracce/s, dipende dal dispositivo)
    - t_S = 5 ÷ 10 ms (valori medi tipici)
  - tempo di rotazione: per posizionarla all'inizio del settore:
    - t_R = 5.6 ms (valore medio per HD "lenti", a 5400 rpm)
    - t_R = 3 ms (valore medio per HD "veloci", a 10000 rpm)
    - t_R = 100 ms (valore medio per floppy, a 300 rpm)
    - (valori medi di t_R = la metà del tempo T_R di una rotazione)
- L'efficienza negli accessi al disco è di grande importanza e ad essa viene dedicata particolare attenzione.

Tempi di accesso al disco

- Il tempo complessivo per l'accesso al disco è la somma di:
  - l'attesa che il dispositivo venga assegnato al processo
  - l'attesa che il canale (se usato da più dischi) sia disponibile
  - il tempo di seek t_S
  - il tempo di rotazione t_R
  - il tempo per il trasferimento dei dati, mentre il settore scorre sotto la testina: per un settore $t_{sett} = T_R/(\#$ settori per traccia):



Esempio di tempi di accesso

- Problema: si debba leggere un file che occupa 2560 settori (1.3 MB) da un disco con le seguenti caratteristiche:
  - tempo di seek medio $t_{Sm} = 10$ ms,
  - tempo di rotazione medio $t_{Rm} = 3$ ms (10000 rpm, $T_R = 6$ ms),
  - 512 byte/settore,
  - 320 settori/traccia
  - tempo di trasferimento /settore $t_{sett} = 6$ ms/320 = 18.75 μs.

Confronto tra tempi di accesso

Il file si trova in settori e tracce contigui:

- si devono leggere 2560/320 = 8 tracce complete,
- tempo medio di lettura della prima traccia $t_{Lprima} = t_{Sm} + t_{Rm} + T_R$
- t_{Lprima} = 10 ms + 3 ms + 6 ms = 19 ms
- per le rimanenti 7 tracce (solo il tempo di rotazione): 7 x T_R,
- tempo totale = 19 ms + 7 x 6 ms = 61 ms.

Il file si trova in settori sparsi in modo casuale sulle superfici del disco:

- si devono leggere 2560 settori,
- tempo medio di lettura di ciascun settore $t_{Lsett} = t_{Sm} + t_{Rm} + t_{sett}$,
- t_{Lsett} = 10 ms + 3 ms + 0.019 ms = 13.019 ms
- tempo totale = 2560 x 13.019 = 33328 ms = 33.3 s

Il tempo richiesto è più di 500 volte più lungo!

Schedulazione degli accessi al disco

- A causa dei tempi di seek, l'accesso casuale al disco è molto più lento di quello sequenziale
- Nei sistemi multitask, le richieste di accesso al disco sono di tipo casuale
- È necessaria una politica che trasformi le sequenze di richieste casuali, in sequenze di accessi ordinati in modo tale da ridurre i tempi di seek

- Politiche di disk scheduling:
  - FIFO
  - SSTF (Shortest Seek Time First)
  - SCAN (C-SCAN)
  - LOOK (C-LOOK)

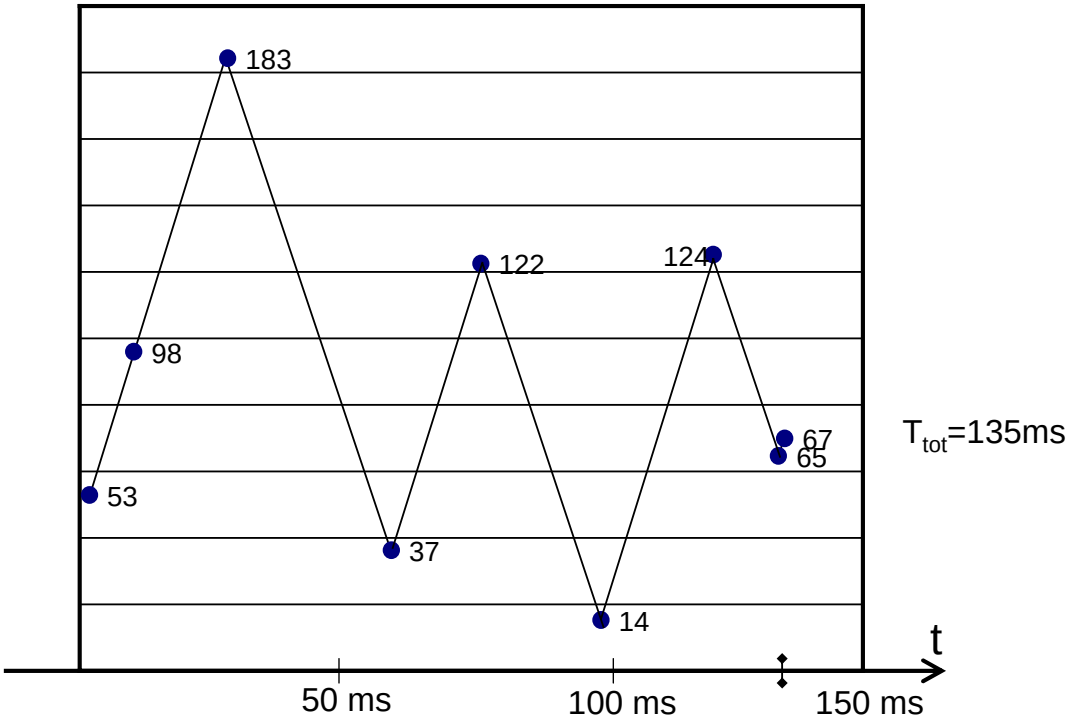
Politiche di disk scheduling

- scheduling FIFO: le richieste di accesso vengono evase nell'ordine con cui sono pervenute
- scheduling SSTF: viene scelta ogni volta come richiesta successiva da evadere, quella situata sulla traccia più vicina alla posizione attuale della testina
- scheduling SCAN: le richieste vengono evase secondo l'ordine con cui vengono incontrate dalla testina, che si sposta nella stessa direzione da una estremità all'altra del disco e inverte la direzione solo quando ha raggiunto l'estremità
- scheduling LOOK: l'ordine di evasione delle richieste è lo stesso del caso SCAN, ma la testina inverte la direzione quando ha raggiunto l'ultima richiesta in quella direzione
- scheduling C-SCAN e C-LOOK: le richieste vengono evase solo mentre la testina sta procedendo in una direzione (il ritorno all'altra estremità avviene senza accessi al disco)

Scheduling FIFO

- Si supponga che, in un dato istante, si sia formata una coda di richieste di accesso al disco, relative a settori situati nelle seguenti tracce (le richieste siano pervenute nel seguente ordine temporale e siano ancora da evadere):
  - tracce richieste: 53, 98, 183, 37, 122, 14, 124, 65, 67
- Si supponga che la testina si trovi inizialmente sulla traccia 53
- Con lo scheduling FIFO vengono evase nell'ordine di arrivo, come rappresentato nella figura seguente
- Le prestazioni sono vicine a quelle dell'accesso casuale (cioè molto scarse).

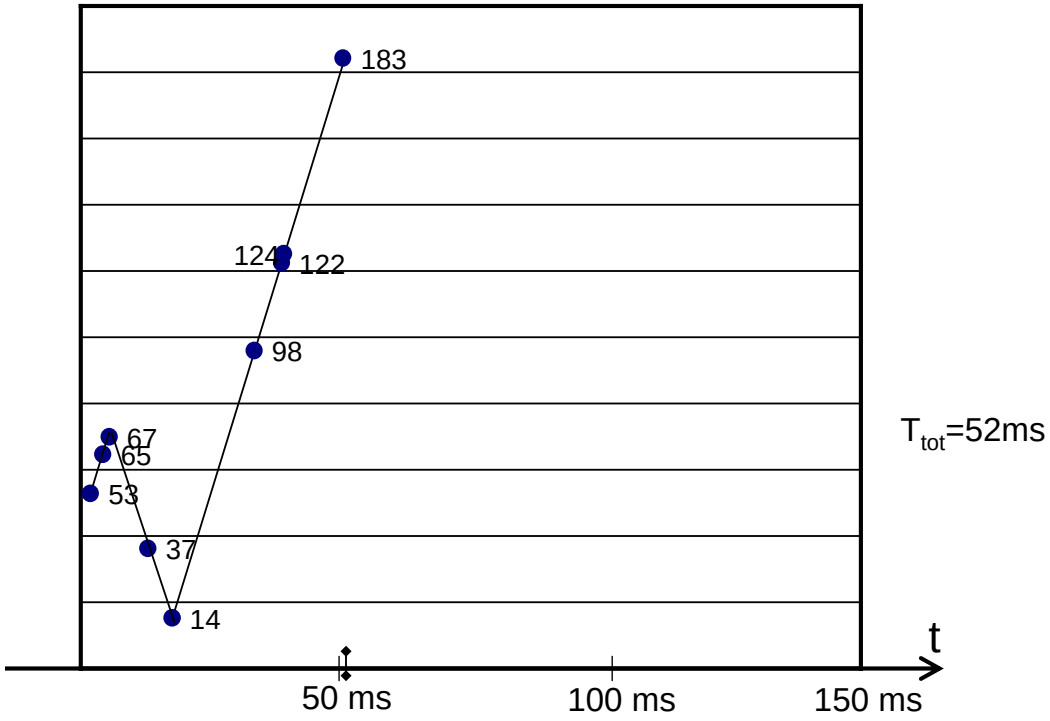
Scheduling FIFO [2]



Scheduling SSTF

- Con lo scheduling SSTF (Shortest Seek Time First), l'ordine di evasione delle richieste porta ogni volta sulla traccia più vicina (con scelta casuale in caso di richieste equidistanti),
- Il tempo complessivo è decisamente più breve (e quindi le prestazioni sono migliori) del caso FIFO
- Però, se continuano a pervenire richieste per tracce vicine alla attuale, può provocare "starvation" delle richieste più lontane.

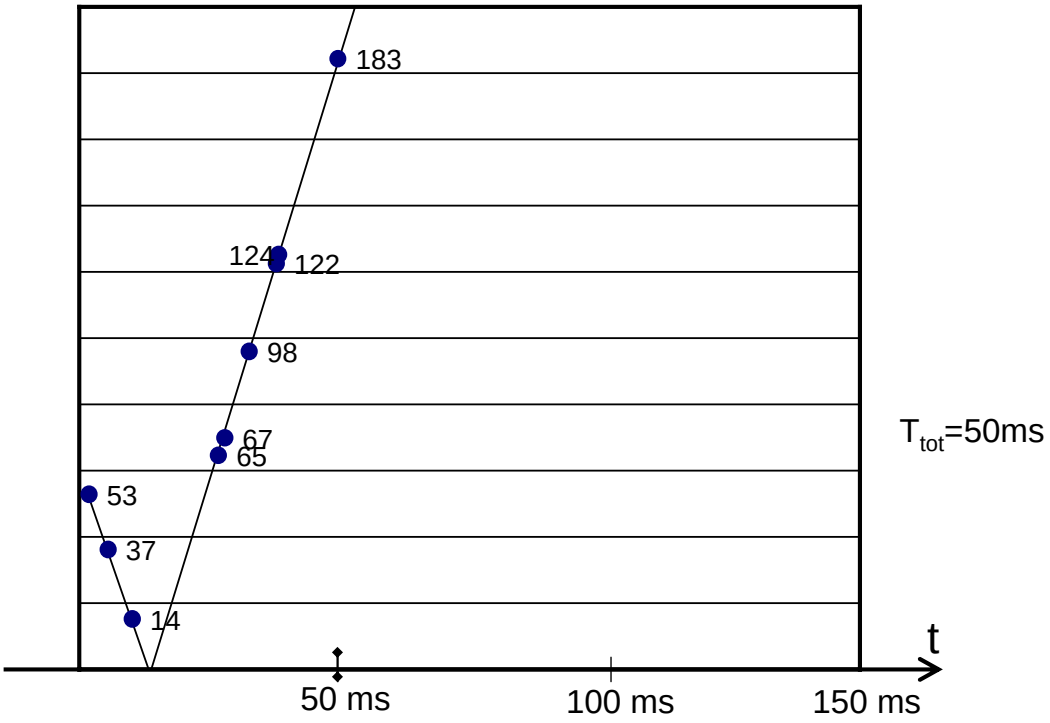
Scheduling SSTF [2]



Scheduling SCAN

- Con lo scheduling SCAN le richieste vengono evase secondo l'ordine con cui vengono incontrate dalla testina, che si sposta nella stessa direzione da una estremità all'altra del disco e inverte la direzione solo quando ha raggiunto l'estremità,
- Lo scheduling SCAN evita il rischio di starvation ed ha una buona efficienza, simile a quella del metodo SSTF,
- Questa politica di scheduling però non è equa:
  - favorisce le richieste che interessano le tracce situate vicino alle estremità delle superfici del disco (vengono scandite due volte, avanti e indietro, in tempi ravvicinati)
  - favorisce anche le richieste arrivate per ultime
- Il primo dei due problemi è risolto con la variante C-SCAN, il secondo con la variante N-step SCAN

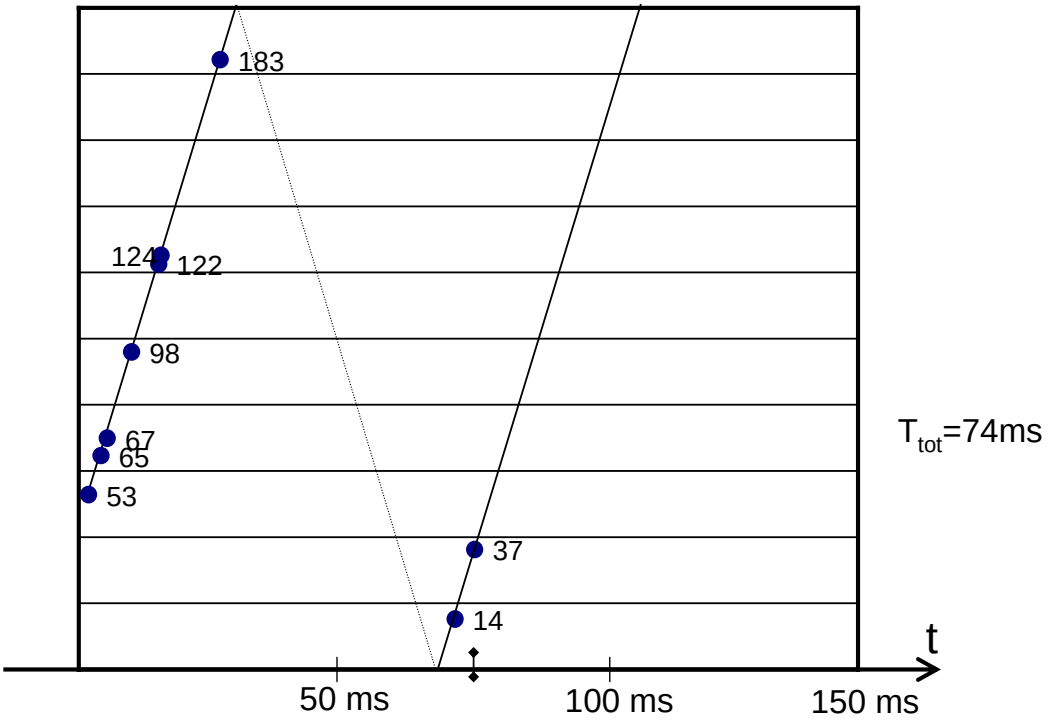
Scheduling SCAN [2]



Scheduling C_SCAN

- Con lo scheduling C-SCAN le richieste vengono evase solo mentre la testina sta procedendo in una direzione (il ritorno all'altra estremità avviene senza accessi al disco)
- Questa politica è più equa del caso SCAN (ogni zona del disco è scandita con la stessa periodicità), ma è anche meno efficiente

Scheduling C_SCAN [2]



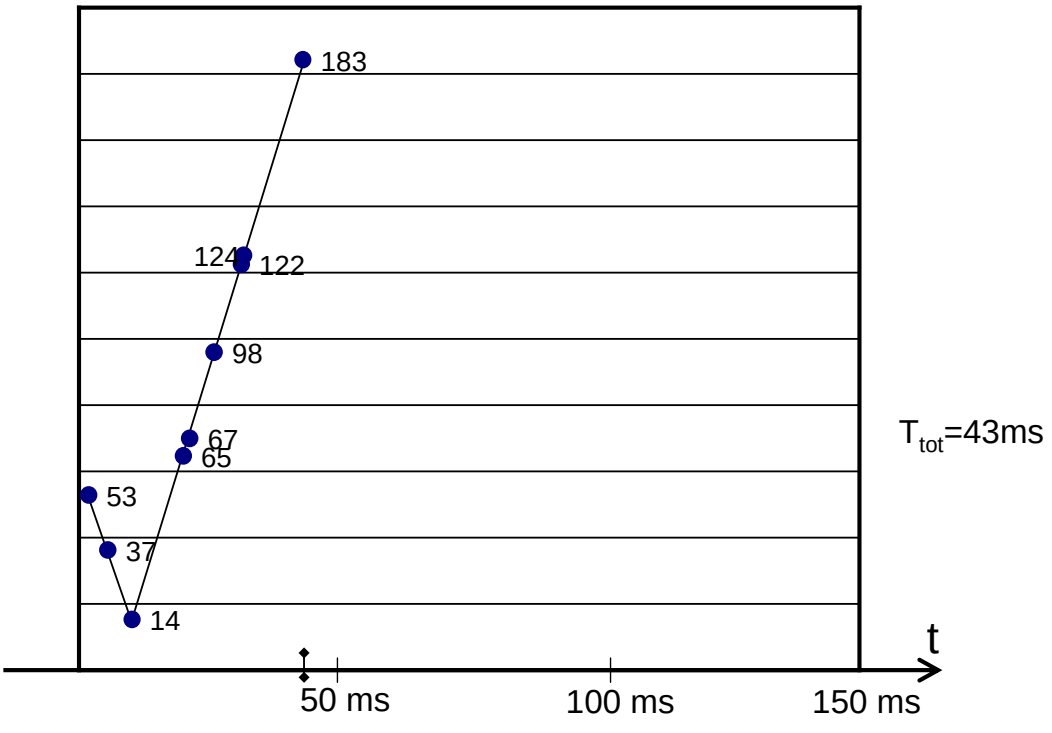
Variante N-step SCAN

- È una variante dello SCAN intesa ad evitare di privilegiare le richieste arrivate per ultime (e di penalizzare le altre)
- Differisce dallo scheduling SCAN per il fatto che:
  - le richieste sono ripartite in sottocode di lunghezza N
  - viene elaborata, con modo SCAN, una sottocoda alla volta
  - se N = 1 si ricade nel caso FIFO
  - se N è grande le prestazioni sono simili allo SCAN
  - le nuove richieste sono inserite in una sottocoda diversa da quella in elaborazione, per cui non possono passare avanti
- Spesso si usano due sottocode (FSCAN):
  - le nuove richieste vengono inserite in una, mentre è servita l'altra.

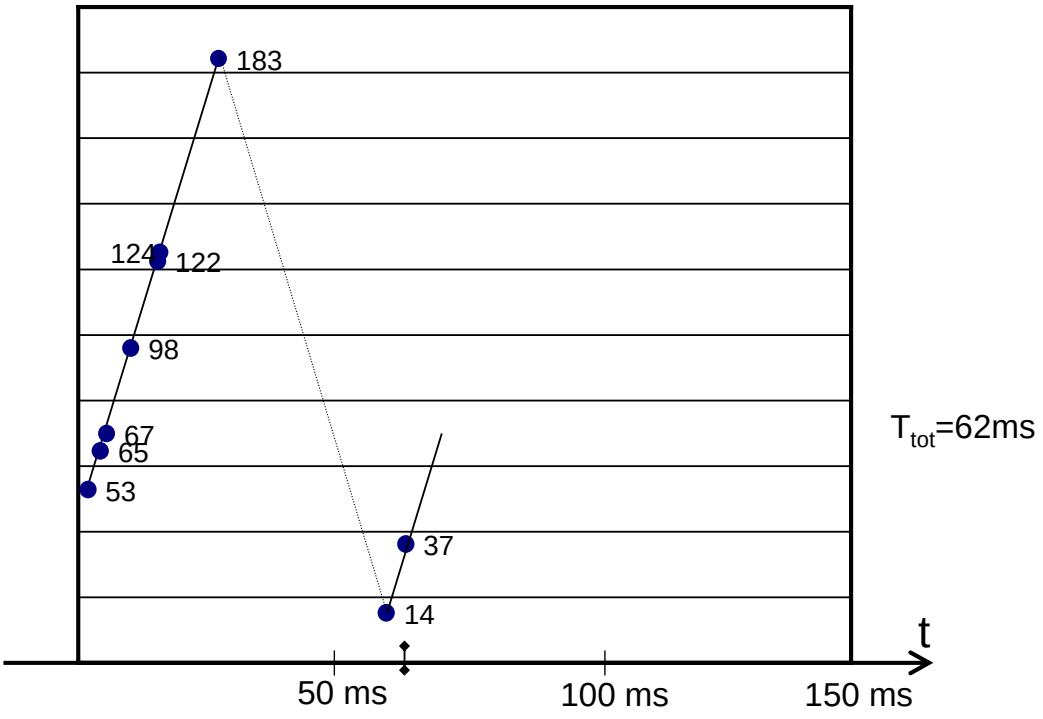
Scheduling LOOK e C-LOOK

- Differiscono dagli scheduling SCAN e C-SCAN solo per il fatto che evitano di procedere in una direzione se non c'è almeno una richiesta in quella direzione
- Pertanto presentano le stesse caratteristiche dello SCAN e, rispettivamente, dello C-SCAN, ma risultano più efficienti

Scheduling LOOK



Scheduling C_LOOK



Prestazioni e affidabilità dei dischi

Le prestazioni (velocità, capacità) di un singolo disco sono **limitate** dalla tecnologia usata:

- per aumentare le prestazioni si può pensare di far operare **più dischi in parallelo**:
  - per servire in parallelo più richieste,
  - per servire in parallelo una singola richiesta (se i dati sono distribuiti su più dischi).

Anche **l'affidabilità** dei dischi è **limitata**,

- con un numero elevato di dischi poco costosi, si può **introdurre ridondanza** per aumentare la affidabilità,
  - per rilevare e/o correggere eventuali errori.

Sistemi RAID

RAID – Redundant Array of Inexpensive Disks

è uno standard per memorizzare dati su più dischi:

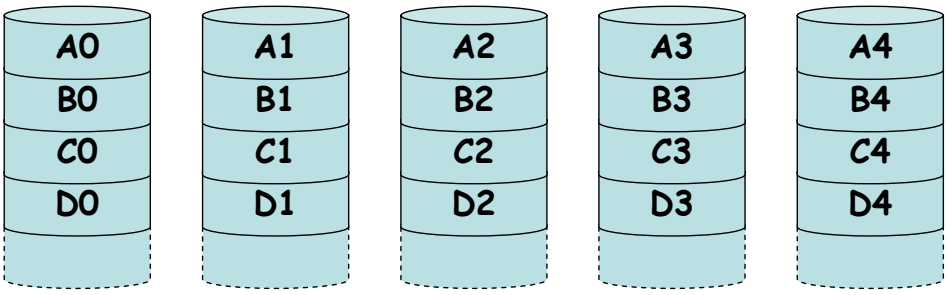
- un insieme di dischi fisici viene visto dal SO come un disco logico unico (**meno costoso, più affidabile**)
- i dati sono **distribuiti su più dischi fisici**,
- si sfrutta l'elevata capacità per memorizzare informazioni di **ridondanza** (parità) che consentono di recuperare i dati in caso di guasto;

Lo standard RAID prevede 7 livelli diversi (da 0 a 6), che indicano organizzazioni diverse con diverse funzionalità;

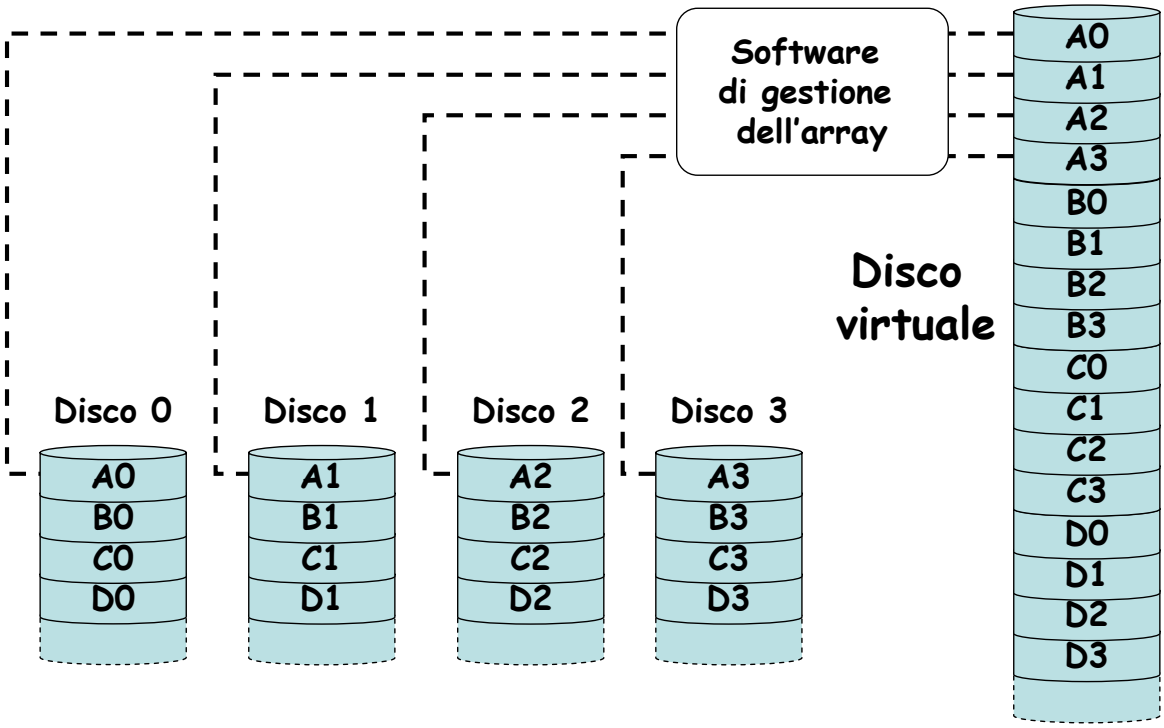
RAID 2 e 4 non sono usati nei sistemi commerciali

RAID 0 (non ridondante)

- Inteso solo a migliorare le prestazioni a basso costo
- Usa la tecnica di **disk striping**: i dati di ciascun file sono suddivisi in stripe (strisce: che possono essere byte, settori o blocchi)
- Stripe consecutivi sono collocati, a rotazione, su dischi diversi
- Una richiesta di I/O è gestita **accedendo in parallelo** ai diversi **stripe collocati su dischi diversi**



RAID 0 (disco logico ⇒ disco fisico)



RAID 0 (prestazioni)

Per applicazioni che richiedono **alte velocità di trasferimento**:

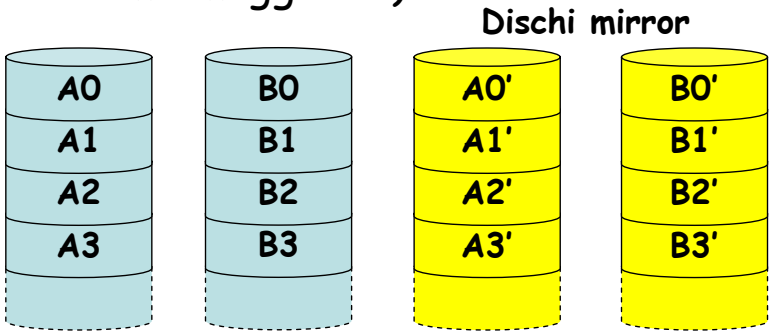
- serve un system bus veloce;
- conviene avere **stripe di dimensioni inferiori alle richieste**, per gestire queste ultime in parallelo su più dischi.

Per applicazioni che richiedono **tempi di risposta brevi**, ovvero con una **elevata frequenza delle richieste** (sistemi interattivi transazionali):

- le richieste sono tipicamente numerose;
- una richiesta riguarda pochi dati: i tempi di I/O sono dominati dai tempi di seek e di rotazione;
- conviene avere **stripe di dimensioni superiori alle richieste**, così ciascuna richiesta è soddisfatta da un singolo disco.

RAID 1 (mirroring)

- La ridondanza consiste nella **duplicazione dei dati** (mirroring)
- Le operazioni di lettura di uno stripe sono effettuate su uno qualsiasi dei due dischi che lo contengono
- Le **operazioni di scrittura** di uno stripe sono effettuate in parallelo sui due dischi (va aggiornata la copia) e quindi limitate dalla velocità del più lento dei due (seek time maggiore)

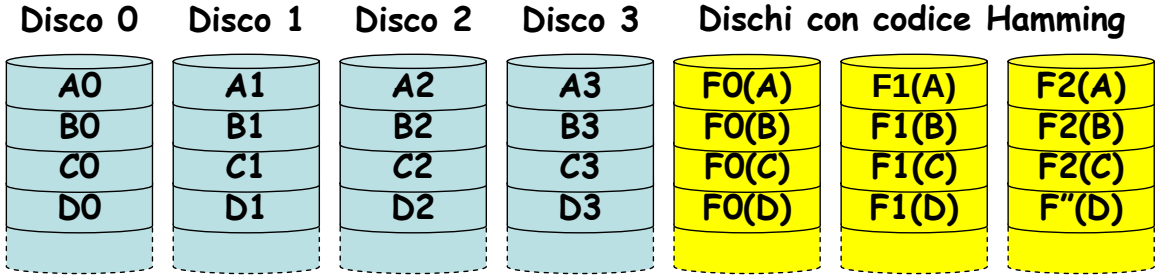


RAID 1 (prestazioni)

- In caso di guasto di un disco, l'esecuzione del processo può continuare usando la copia:
  - è una forma di **backup in tempo reale**
  - **raddoppia il costo**
- Se le richieste sono in **maggioranza di lettura**, presenta anche vantaggi rispetto al RAID 0:
  - la velocità di trasferimento può raddoppiare: le operazioni di lettura possono coinvolgere, in parallelo, un disco e (altri stripe del)la sua copia
  - la frequenza delle richieste può raddoppiare, c'è il doppio di dischi disponibili per gli accessi
- Se le richieste sono di scrittura, le prestazioni sono le stesse del RAID 0

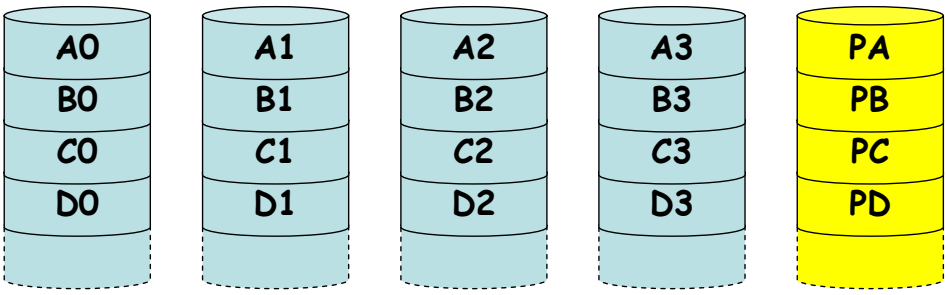
RAID 2 (codice di Hamming)

- Usa un **ECC** (Error Correcting Code) di N bit (**codice di Hamming**) per rilevare/correggere gli errori negli stripe registrati in posizioni corrispondenti
- Servono **N dischi in più** per contenere questo codice
- Consente **la correzione di singoli bit** errati e il rilevamento di errori su più bit
- RAID 2 non è usato nei sistemi commerciali



RAID 3 (bit di parità)

- La dimensione degli stripe è piccola (**byte** o word)
- L'accesso parallelo ai dischi **è sincronizzato** (le testine di tutti i dischi si trovano nella stessa posizione)
- Vi è un disco in più, che contiene solo i **bit di parità** calcolati sui bit nella medesima posizione di tutti gli altri dischi
- Se un disco va fuori uso, il suo contenuto può essere **ricostruito a partire dagli altri dischi**



RAID 3 (prestazioni)

In un array di 5 dischi, in cui X0 ÷ X3 contengono i dati e X4 i bit di parità, i bit di X4 sono calcolati con:

- **X4(i) = X3(i) ⊕ X2(i) ⊕ X1(i) ⊕ X0(i)**

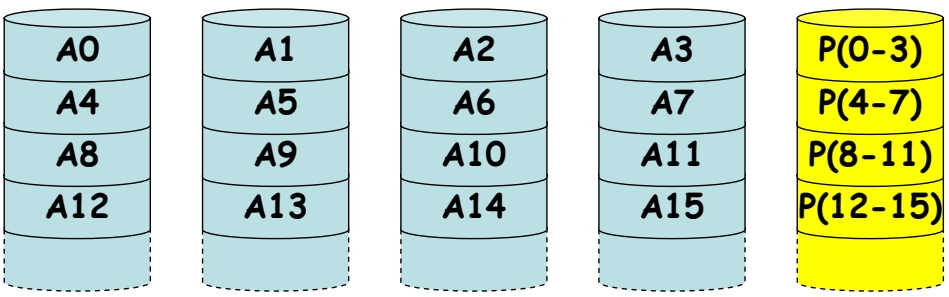
In caso di guasto del disco X1, il suo contenuto può essere **ricostruito al volo** durante le operazioni di lettura e scrittura:

- **X1(i) = X4(i) ⊕ X3(i) ⊕ X2(i) ⊕ X0(i)**

- Può essere servita solo una richiesta alla volta, per cui la frequenza delle richieste non è molto alta
- Gli stripe di piccole dimensioni consentono un elevato livello di parallelismo (tutti i dischi sono interessati, in parallelo)

RAID 4 (parità a livello di blocco)

- Ciascun disco opera indipendentemente (le testine non sono nella stessa posizione), per cui possono essere servite **più richieste in parallelo**
- Pertanto il sistema è adatto ad applicazioni che richiedono **frequenze elevate** di richieste
- Gli stripe sono di **grandi dimensioni** (blocchi di byte)
- Per ciascun blocco viene creato un **blocco contenente i bit di parità**, memorizzato su un **disco di parità**



RAID 4 (prestazioni)

- In un array di 5 dischi, in cui X0 ÷ X3 contengono i dati e X4 i bit di parità, i bit di X4 sono calcolati con:

  X4(i) = X3(i) ⊕ X2(i) ⊕ X1(i) ⊕ X0(i)

- In caso di modifica dello stripe nel disco X1 (per una **operazione di scrittura**), vanno modificati X1 e X4:

  X4'(i) = X3(i) ⊕ X2(i) ⊕ X1'(i) ⊕ X0(i)

- Non essendo i dischi sincronizzati, conviene evitare di leggerli tutti 4 (e poi scrivere su X4):

  X4'(i) = X3(i) ⊕ X2(i) ⊕ X1'(i) ⊕ X0(i) ⊕ X1(i)⊕X1(i)

  X4'(i) = X4(i) ⊕ X1'(i) ⊕ X1(i)

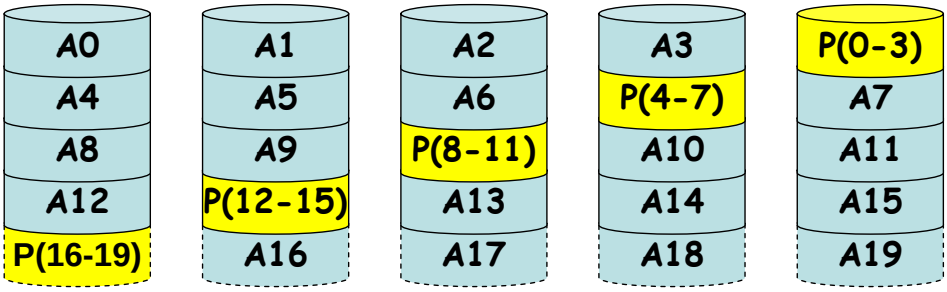
RAID 4 (prestazioni)

X4'(i) = X4(i) ⊕ X1'(i) ⊕ X1(i)

- Un'operazione di write su uno strip richiede:
  - di leggere i vecchi stripe dati e di parità (X1 e X4), prima della scrittura;
  - di scrivere i nuovi stripe dati e di parità (X1' e X4').
- Ogni operazione di **scrittura richiede 4 accessi**: due di lettura e due di scrittura
- Il disco di parità **subisce 2 accessi** per ogni operazione di scrittura: può essere un **collo di bottiglia!**
- RAID 4 non è usato nei sistemi commerciali\

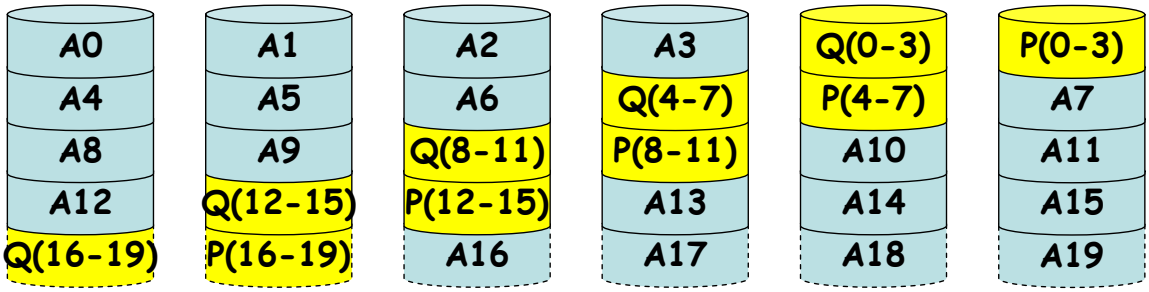
RAID 5 (parità distribuita)

- A differenza del RAID 4, gli **stripe di parità** sono **distribuiti su tutti i dischi** (tipicamente secondo uno schema circolare)
- Si evita il collo di bottiglia sul disco di parità segnalato per il livello 4.



RAID 6 (parità doppia)

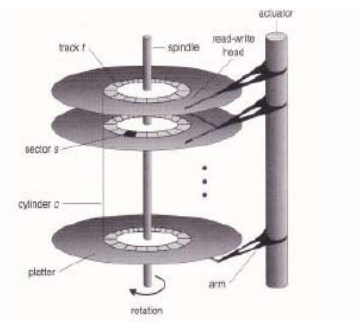
- Aumenta la ridondanza rispetto ai livelli precedenti
- Usa **due diversi algoritmi di parità**: P (basato sull'OR esclusivo usato nei livelli 4 e 5) e Q (un algoritmo di check diverso)
- I risultati dei due algoritmi sono memorizzati in due blocchi diversi su **due dischi diversi**



RAID 6 (prestazioni)

- Richiede **due dischi in più**
- Consente di rigenerare i dati anche nel caso di **due dischi fuori uso**
- Ciascuna operazione di scrittura deve aggiornare anche due blocchi di parità
- Adatto per le applicazioni in cui è importante una elevata affidabilità dei dischi

Fine



Gestione dei dischi