Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2019-2020

Pietro Frasca

Lezione 20

Giovedì 12-12-2019

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 del settore** (**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:

  Tts=6 ms /320 = 0,01875 ms che è circa 19 µS

- 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.

Esempio

- Per mostrare l'importanza del modo di memorizzare i dati su disco consideriamo il caso di memorizzazione di un file di 320 KB:
  - 1) Memorizzazione su due tracce contigue.
  - 2) Memorizzazione su tracce e settori sparsi. (esempio di alta frammentazione del disco)

Memorizzazione su due tracce contigue

- Per il caso 1 il tempo **Tmt** si calcola:

  ns=file_size/sec_size=320*1024/512=640 settori

  se il file è allocato in due tracce adiacenti si ha che il tempo necessario per leggere le tracce è dato:

  prima traccia

  Tmt1=Tseek+Trot/2+320*Tts=5.2+3+0.019*320=14.28 ms

  seconda traccia

  Tmt2=Tseek_min+Trot/2+320*Tts=0.6+3+0.019*320=9.68 ms

Tmt=Tmt1+Tmt2=14.28+9.68=23.96 ms

Memorizzazione su tracce e settori sparsi

- Nel caso 2 (settori sparsi) si ha:

  Tmt=(Tseek+Trot/2+Tts)*640 = (5.2 +3+0.019)*640=5260.16 ms

  quindi il tempo aumenta di un fattore

  f=5260.16/23.96=219.54

Esempio

- Per mostrare l'importanza del modo di memorizzare i dati su disco consideriamo il caso di memorizzazione di un file di 320 KB:
  - 1) Memorizzazione su due tracce contigue.
  - 2) Memorizzazione su tracce e settori sparsi. (esempio di alta frammentazione del disco)

Memorizzazione su due tracce contigue

- Per il caso 1 il tempo **Tmt** si calcola:

  ns=file_size/sec_size=320*1024/512=640 settori

  se il file è allocato in due tracce adiacenti si ha che il tempo necessario per leggere le tracce è dato:

  prima traccia

  Tmt1=Tseek+Trot/2+320*Tts=5.2+3+0.019*320=14.28 ms

  seconda traccia

  Tmt2=Tseek_min+Trot/2+320*Tts=0.6+3+0.019*320=9.68 ms

Tmt=Tmt1+Tmt2=14.28+9.68=23.96 ms

Memorizzazione su tracce e settori sparsi

- Nel caso 2 (settori sparsi) si ha:

  Tmt=(Tseek+Trot/2+Tts)*640 = (5.2 +3+0.019)*640=5260.16 ms

  quindi il tempo aumenta di un fattore

  f=5260.16/23.96=219.54

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

Scheduling del disco

- In un sistema multiprogrammato con molti processi attivi, la coda del disco spesso può avere diverse richieste in sospeso. Così, quando una richiesta è completata, il firmware del HDD sceglie quale richiesta in attesa conviene servire prima. Per tale scelta si ricorre a diversi algoritmi di scheduling del disco.
- Esamineremo tre noti algoritmi di pianificazione del disco: FCFS, SSTF e SCAN.

Scheduling in ordine di arrivo - FCFS

- L'algoritmo più semplice di scheduling del disco è, naturalmente, ***FCFS (First Came First Served)***, basato sull'ordine di arrivo. Questo algoritmo ha un comportamento equo, ma in genere non fornisce il servizio più veloce.
- Consideriamo, per esempio, una coda di richieste per un disco con relative a settori sui seguenti cilindri e in questo ordine: 14, 40, 23, 47, 7.

- Supponiamo, inoltre, e che le testine siano correntemente posizionate sul cilindro 20.
- Tipicamente, con FCFS, il percorso che la testine compiono ha una forma altalenante. In questo caso ciascuna testina esegue un percorso totale, misurato in numero di cilindri attraversati, di 113 cilindri.
- Questa pianificazione è schematizzata nella figura seguente

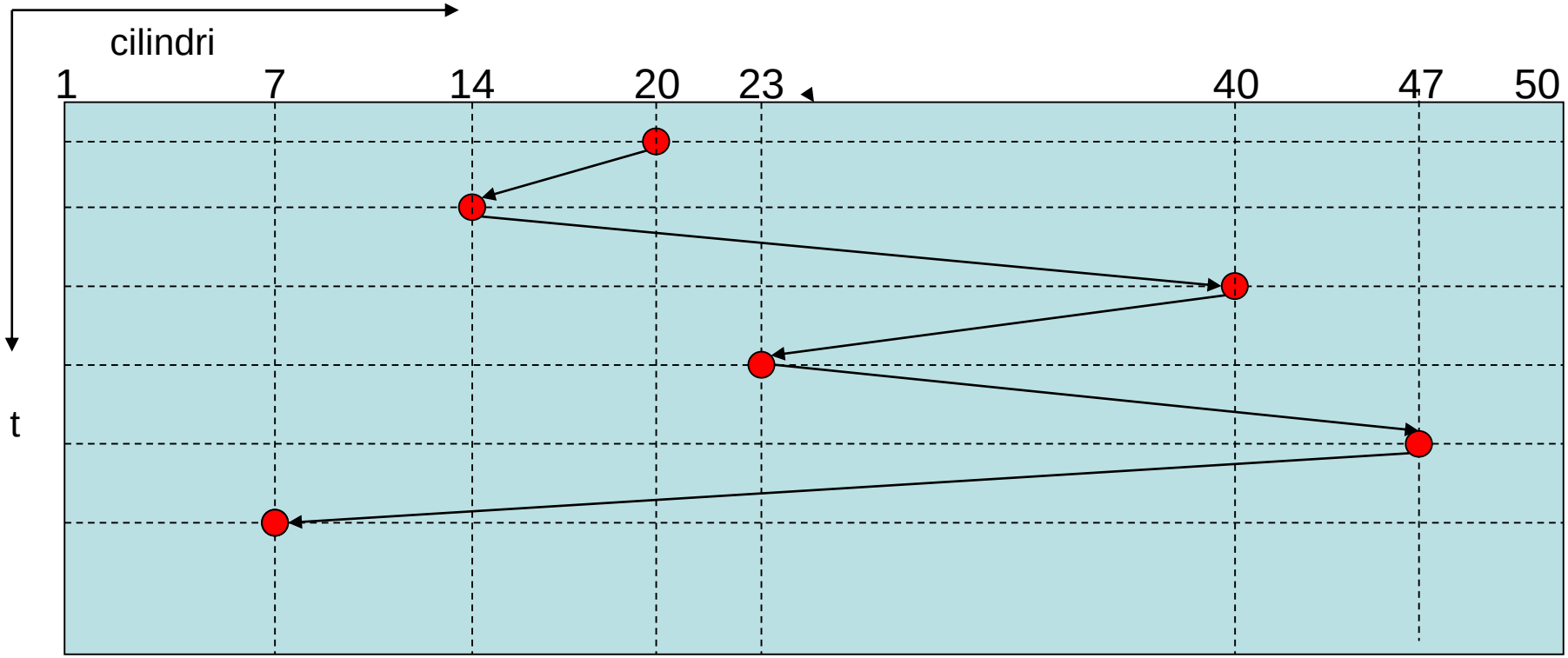
- **Sceduling in ordine di arrivo - FCFS**

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 14, 40, 23, 47, 7

Percorso: 113 cilindri



Scheduling per brevità - SSFT

- L'algoritmo ***SSTF (Short Seek Time First)*** si basa sulla ragionevole evidenza che è più conveniente servire tutte le richieste vicine alla posizione attuale della testina prima di spostarla lontano per servire altre richieste.
- Pertanto, SSTF sceglie la richiesta in sospeso relativa al cilindro più vicino al cilindro sui cui è posizionata attualmente la testina.
- Per il nostro esempio, questo metodo di pianificazione determina un movimento totale della testina di soli 59 cilindri, poco più della metà della distanza percorsa da FCFS.
- Chiaramente, questo algoritmo fornisce un sostanziale miglioramento delle prestazioni.
- SSTF è essenzialmente una forma di scheduling per brevità, come l'algoritmo SJF (Short Job First) descritto nello scheduling della CPU. Come SJF, anche SSFT può causare problemi di attesa indefinita (starvation).

- Infatti, se sono presenti in coda molte richieste per accedere a cilindri tra loro vicini e una richiesta distante da questi la richiesta del cilindro lontano rischia di non essere soddisfatta per un tempo eccessivo. Questo scenario diventa sempre più probabile se la coda delle richieste si allunga.

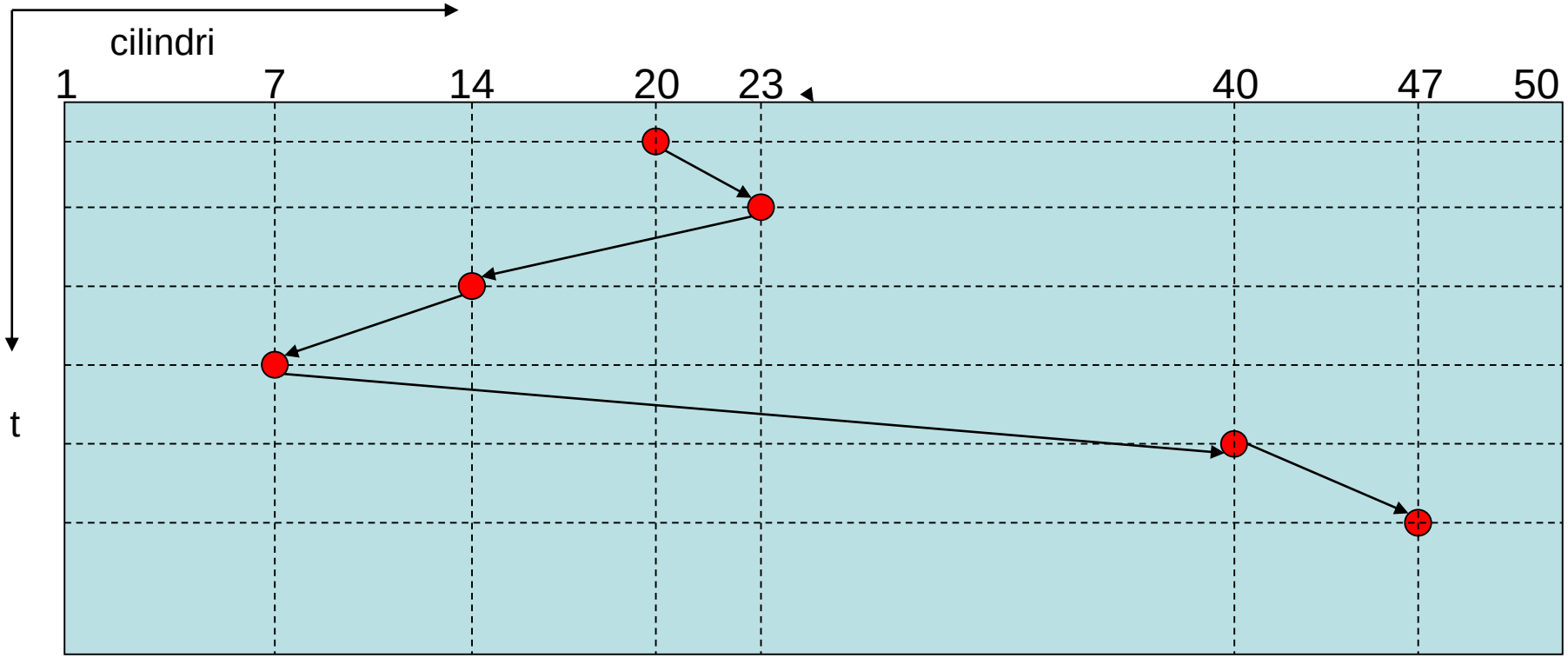
- **Sceduling per brevità – SSTF (Shortest Seek Time First)**

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 23, 14, 7, 40, 47

Percorso: 59 cilindri



Scheduling per scansione - SCAN

- Con l'algoritmo SCAN, il braccio del disco si muove partendo da un estremità del disco all'altra servendo le richieste durante l'attraversamento dei cilindri.
- Raggiunto il cilindro con numerazione estrema, il braccio è mosso nella direzione opposta. La testina si muove continuamente avanti e indietro attraverso il disco.
- Per questo funzionamento, l'algoritmo SCAN è spesso chiamato algoritmo dell'ascensore, in quanto il braccio del disco si comporta come un ascensore con prenotazione in un edificio.
- Per il nostro esempio il percorso con SCAN è mostrato nella figura seguente. Il percorso è di 53 cilindri.
- Questo algoritmo elimina la possibilità che si verifichi starvation.

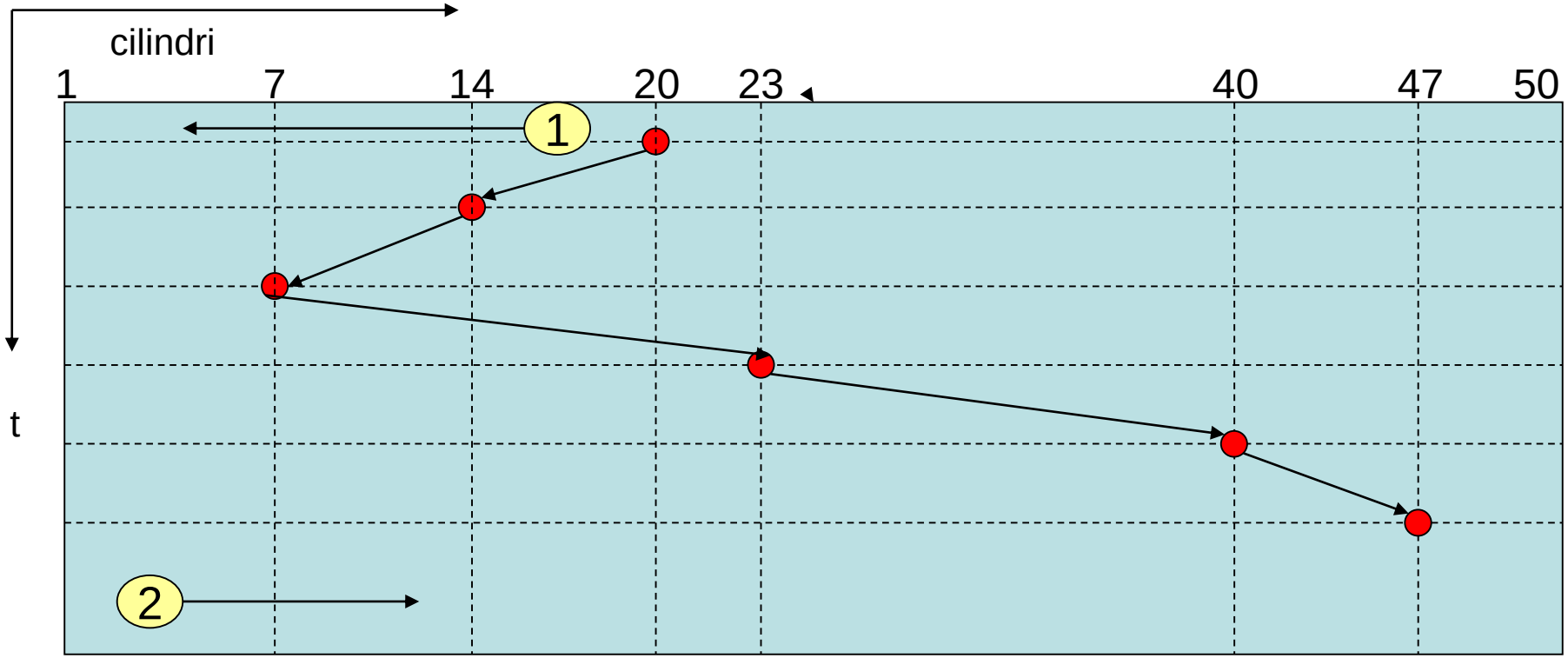
Sceduling per scansione – SCAN

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 14, 7, 23, 40, 47

Percorso 53 cilindri

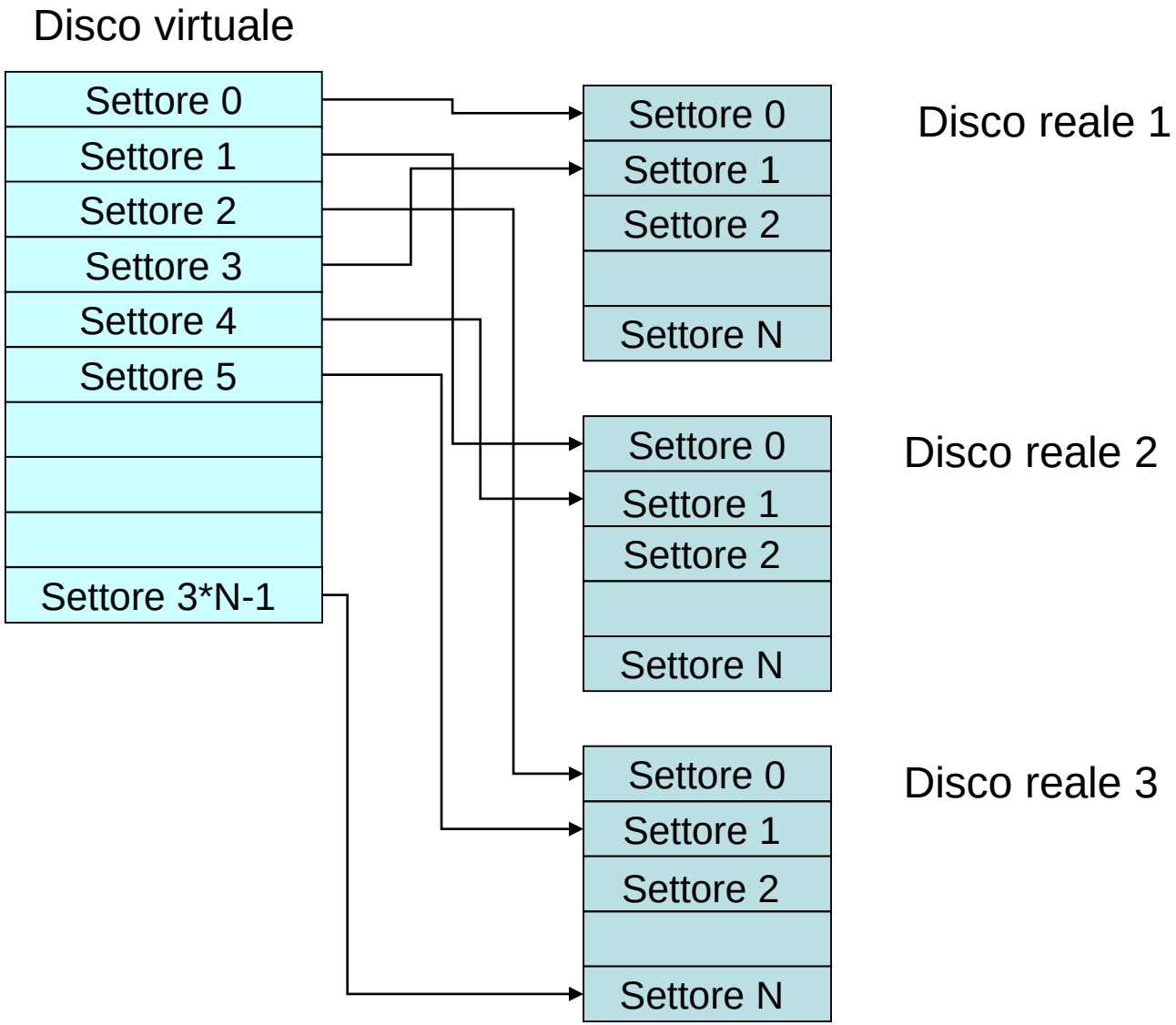


Dischi RAID (Redundancy Array of Independent Disk)

- E’ possibile migliorare le prestazioni dei dischi mediante particolari tecnologie.
- Uno standard molto diffuso è **RAID** (Redundancy Array of Independent Disk), che consente di memorizzare i dati su array di dischi, per migliorare l’affidabilità e la velocità degli accessi.
- Con questa tecnologia, in un sistema con **N** dischi, ciascuno di capacità **disk_size,** è possibile realizzare un disco virtuale di capacità **N*disk_size** e con tempo di accesso notevolmente inferiore.
- Lo standard RAID ha 7 varianti indicate con ***varianti di livello 0,1,2,3,4,5 e 6***. Ogni livello ha caratteristiche diverse, in dipendenza della ridondanza di dati e della velocità di accesso che si vogliono ottenere.

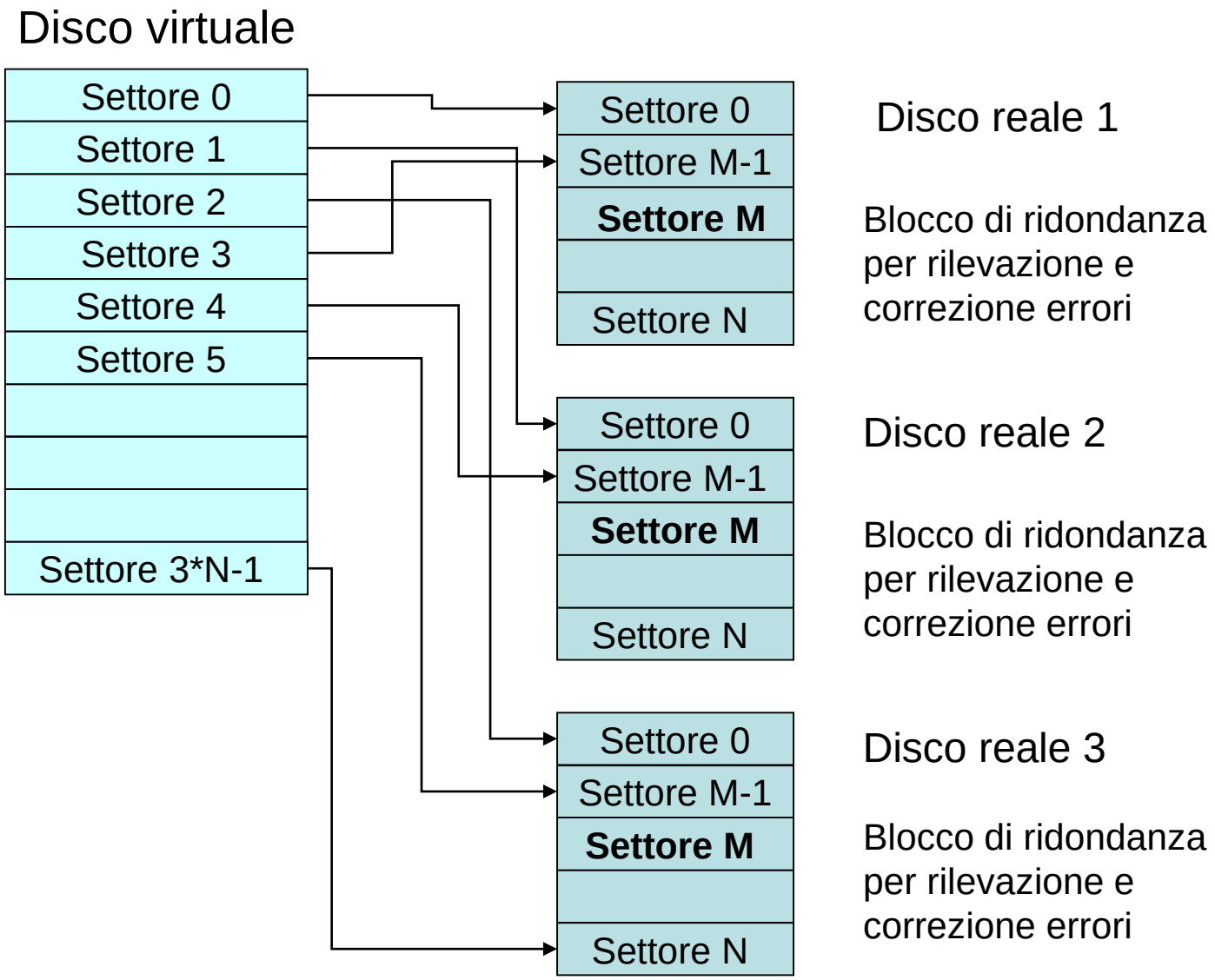
- La variante del livello 0, schema di figura, non produce alcuna ridondanza dei dati.
- La variante di livello 1 (**mirroring**) è come il livello 0 ma con tutti i dischi duplicati (in relazione al caso di figura sarebbero necessari 6 dischi). E'uno schema ottimo sia per quanto riguarda l'affidabilità che la velocità di accesso.
- La presenza delle copie consente di effettuare operazioni di scrittura in parallelo mentre le operazioni di lettura si possono ottenere referenziando il disco che richiede il minor tempo di trasferimento. Questo schema è il più costoso in quanto richiede un numero elevato di dischi.
- Ci sono altre varianti che vanno dai livelli 2 al 6, che differiscono dal livello 1 per il livello di ridondanza dei dati e per le capacità di rilevazione e correzione degli errori, che utilizzano schemi per il controllo di parità o codifica di hamming.

- Dischi RAID. Schema di livello 0 (senza ridondanza dei dati)



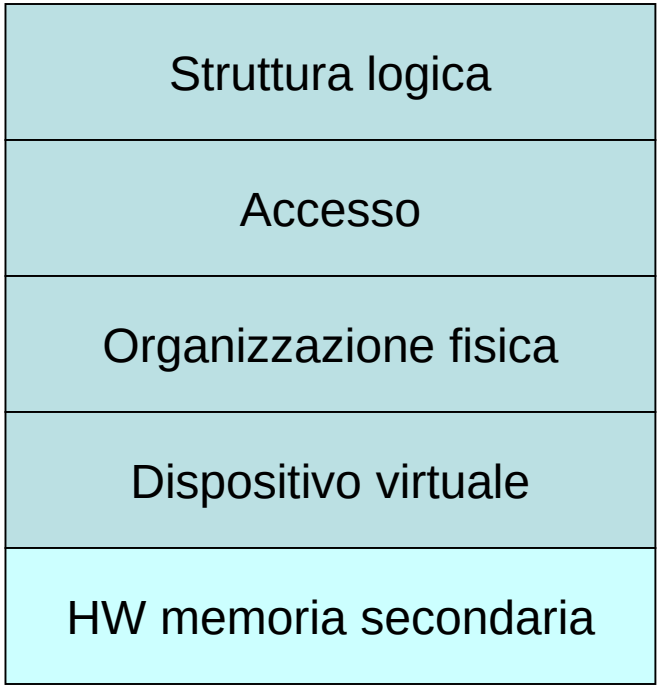
- Il livello 5 ad esempio, presenta un livello di ridondanza nettamente inferiore al mirroring (livello 1). Un determinato numero di blocchi su ogni disco è usato per contenere dati aggiuntivi per il controllo di parità. Ad esempio, nel caso di un array di N dischi, per ogni gruppo di M settori consecutivi, viene calcolato un settore M+1 per contenere bit di parità.

- Dischi RAID. Schema di livello 5



Il file system

- Il file system è la parte del SO che realizza le astrazioni e le tecniche che consentono la rappresentazione, l'archiviazione e l'accesso ai dati memorizzati sulla memoria secondaria.
- La struttura di un file system è formata da vari componenti organizzati in vari livelli:



La struttura logica del file system

La struttura logica consente all'utente di vedere i dati memorizzati sulla memoria secondaria a prescindere dalle caratteristiche dei dispositivi e dalle tecniche di allocazione. I dati sono visti sotto forma di **file e directory (cartelle)**.

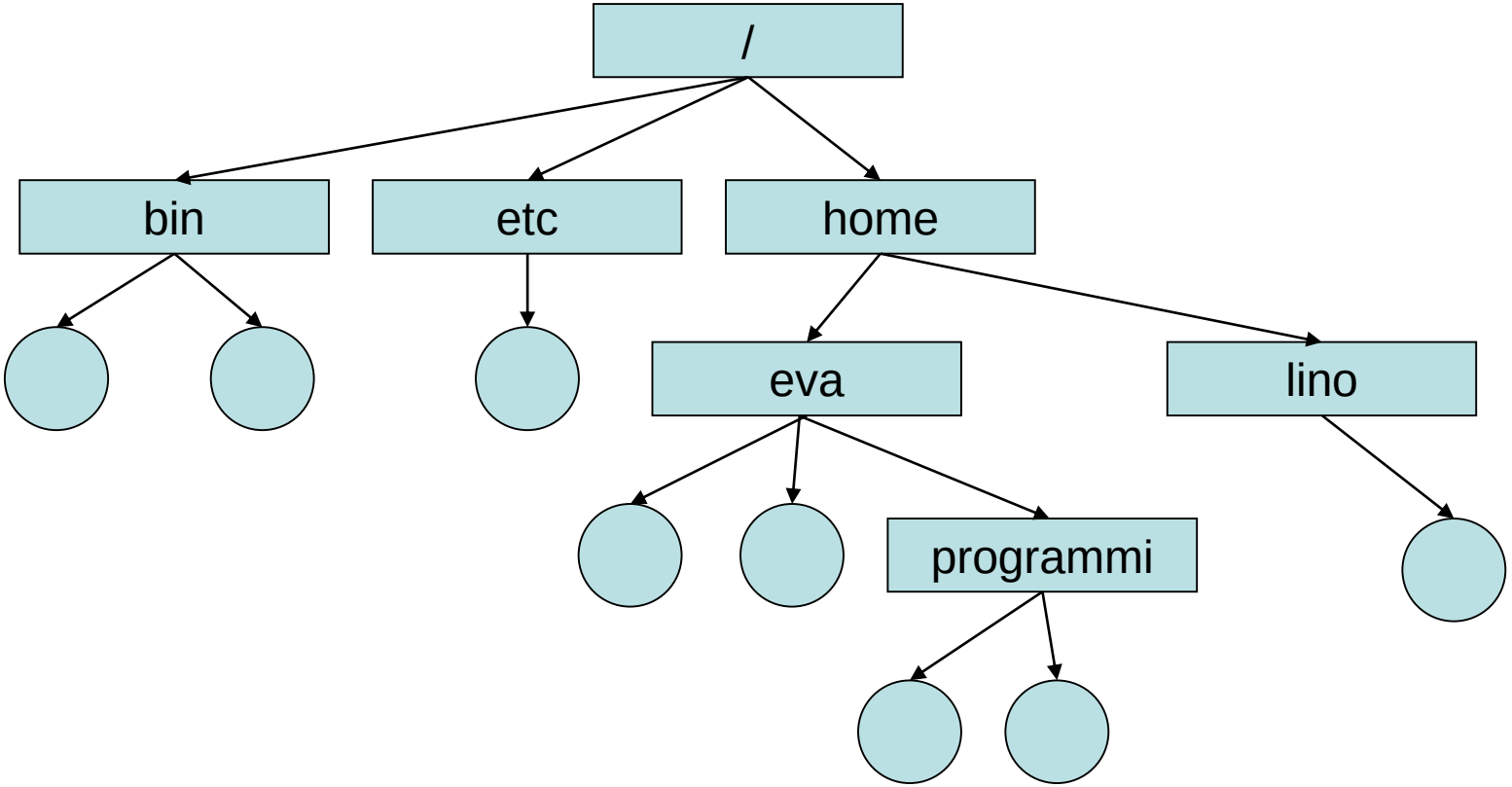
Il file

- il file è l'unità logica di memorizzazione all'interno del file system.
- I file possono contenere informazione di vario genere: testo, programma eseguibile, audio, video, immagini etc.
- Un file è identificato da un **nome** ed è descritto da un insieme di attributi, come ad esempio
  - data ultima modifica
  - dimensione
  - tipo

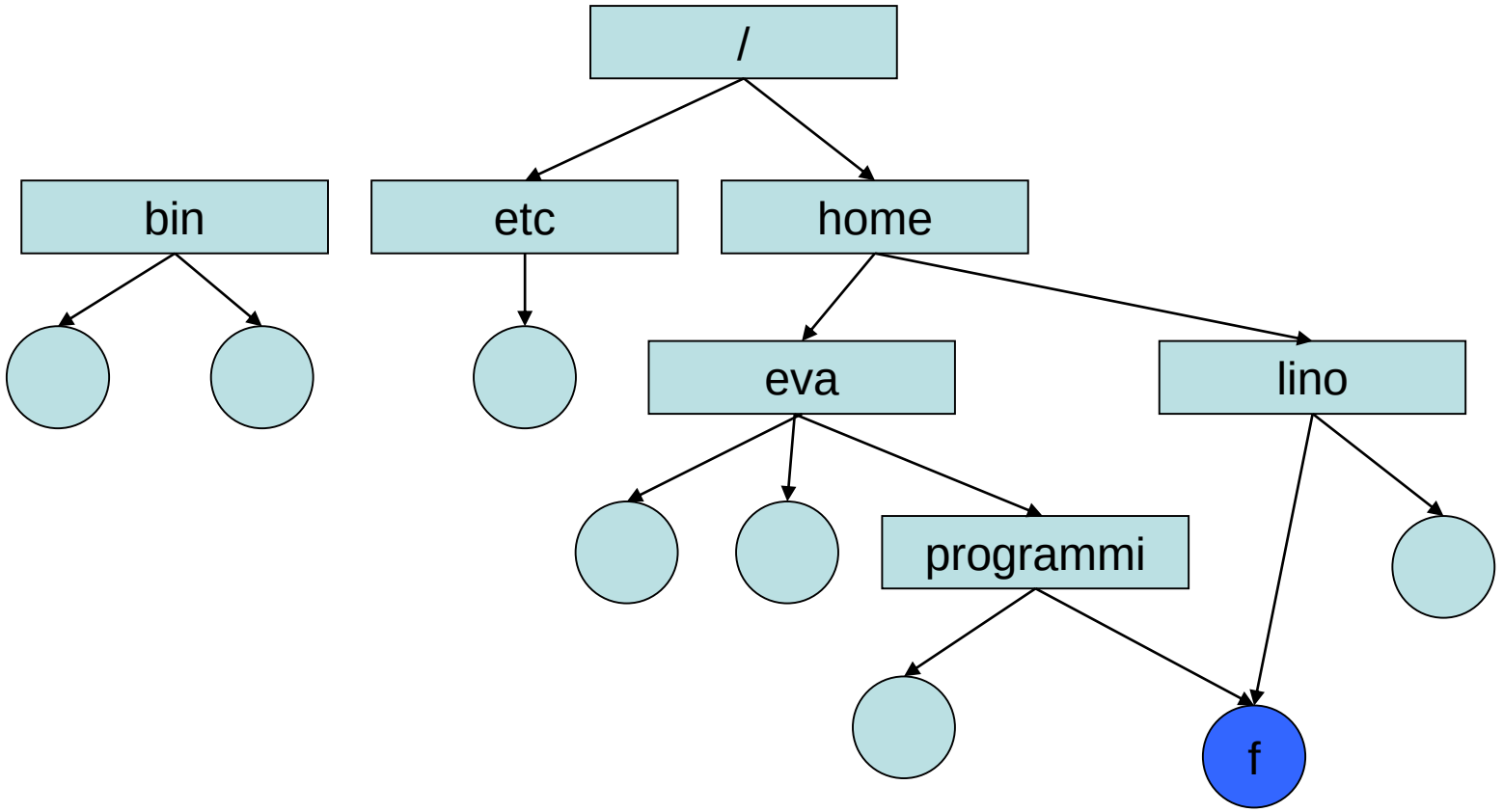
- Nei SO multiutente sono molto importanti:
  - proprietario
  - gruppo
  - diritti di accesso

La directory

- La directory è un’astrazione che consente di raggruppare più file. E’ un contenitore di file.
- Una directory può contenere nel suo interno altre directory (directory nidificate).
- La struttura del file system è quindi un insieme di file e directory.
- Generalmente i file system sono strutturati ad albero o a grafo.



File system con struttura ad albero



File system con struttura a grafo

Gestione della struttura logica del file system

- Il SO offre un insieme di chiamate di sistema per la gestione del file system. Le principali operazioni sono:
  - Creazione, rinomina e cancellazione di directory
  - Creazione, rinomina e cancellazione di file
  - Creazione, rinomina e cancellazione di link
  - Listato di una directory
  - Navigazione del file system
  - Spostamento di file e directory

- E’ possibile eseguire queste operazioni sia mediante interfaccia grafica (GUI) del SO che attraverso *comandi* digitati mediante l’uso di una shell.
- Operazioni più complesse si possono ottenere realizzando script di shell o applicazioni, scritte con un linguaggio, che fanno uso di chiamate di sistema relative al file system.