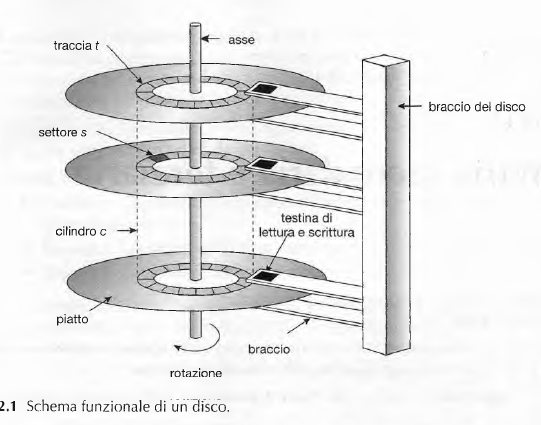
# CAPITOLO 12: MEMORIA SECONDARIA E TERZIARIA

## STRUTTURA DEI DISPOSITIVI DI MEMORIZZAZIONE

## Dischi magnetici

I **dischi magnetici** sono il mezzo fondamentale di memoria secondaria dei moderni sistemi di calcolo. Concettualmente i dischi sono relativamente semplici: i **piatti** hanno una forma piana e rotonda e le 2 superfici ricoperte di materiale magnetico dove le informazioni vengono registrate magneticamente.



Le **testine di lettura e scrittura** sono sospese su ciascuna superficie d’ogni piatto e sono attaccate al braccio del disco che le muove in blocco.

La superficie di un piatto è divisa logicamente in **tracce** circolari a loro volta suddivise in **settori**. L’insieme delle tracce corrispondenti a una posizione del braccio formano un **cilindro**.

In un’unita a disco possono esservi migliaia di cilindri. Quando un disco è in funzione un motore lo fa ruotare ad alta velocità tra 60 e 200 giri al secondo.

L’efficienza di un disco è caratterizzata da due valori: la **velocità di trasferimento,** cioè la velocità con cui i dati fluiscono dall’unità a disco al calcolatore e il **tempo di posizionamento,** (talvolta detto tempo d’accesso diretto), che consiste nel tempo necessario a spostare il braccio del disco in corrispondenza del cilindro desiderato, detto **tempo di ricerca** *(seek time*), e nel tempo necessario affinchè il settore desiderato si porti, tramite la rotazione del disco, sotto la testina, detto **latenza di rotazione.**

In genere i dischi possono trasferire parecchi megabyte di dati al secondo e hanno un tempo di ricerca e una latenza di rotazione di diversi millisecondi.

Poiché le testine di un disco sono sospese su un cuscino d’aria sottilissimo, esiste il pericolo che la testina urti la superficie ; in tal caso si parla di **crollo della testina** e causa la rottura irreversibile del disco.

Un disco può essere **rimovibile**: ciò permette che diversi dischi siano montati secondo le necessità.

I **dischetti** (floppy disk) sono dischi magnetici economici e rimovibili costituiti da un involucro di plastica(che contiene un piatto flessibile) e la capacità di memorizzazione è dell’ordine di 1.44 MB .Per ridurne l'usura , si fa ruotare il dischetto molto più lentamente dei piatti di un hard-disk .

I dischi possono essere collegati attraverso dei fili detti *bus* che possono essere EIDE, SATA, PATA, SCSI, USB.

Il trasferimento dei dati in un bus è eseguito da speciali unità di elaborazione dette *controllori*.

Gli **adattatori** o **controllori di macchina** sono i controllori posti all’estremità relativa al calcolatore del bus.

## 

## Nastri magnetici

I **nastri magnetici** sono stati i primi supporti di memoria secondaria per la loro capacità di immagazzinare una grande quantità di dati, ma la loro scarsa velocità e la loro natura di dispositivi ad **accesso sequenziale** li rende inadatti per essere utilizzati come memorie secondarie.

Gli usi principali dei nastri sono la creazione di copie di riserva dei dati *(backup),* la registrazione di dati poco usati e il trasferimento di informazioni tra diversi sistemi di calcolo.

Il nastro è avvolto in *bobine* e scorre su una *testina* dove i dati sono salvati magneticamente, ma il posizionamento dei dati sotto la testina può richiedere diverso tempo, dovendo necessariamente scorrere tutto il nastro fino al raggiungimento della posizione desiderata.

Le capacità dei nastri variano tra 20 e 200GB. Alcune unità sono dotate di funzionalità di compressione dei dati che permettono di raddoppiare la capacità effettiva. Le unità a nastro e i loro driver sono solitamente classificate per larghezza. Alcune unità prendono il nome dalla tecnologia su cui si fondano.

## STRUTTURA DEI DISCHI

I moderni dischi sono considerati come grandi array monodimensionali di blocchi logici(o cluster), dove un **blocco logico** è la minima unita di trasferimento. La dimensione di un blocco logico è di solito di 512 byte sebbene alcuni dischi si possano formattare a basso livello allo scopo di ottener una diversa dimensione dei blocchi logici. L’array monodimensionale di blocchi logici corrisponde in modo sequenziale ai settori del

disco.

Sfruttando questa corrispondenza almeno in teoria sarebbe possibile trasformare gli indirizzi logici in indirizzi fisici costituiti da numero di cilindro, traccia, settore.

In pratica vi sono 2 motivi che rendono difficile quest’operazione ; in primo luogo la maggior parte dei dischi contiene settori difettosi, in secondo luogo il numero di settori per traccia in certe unità a disco non è costante.

Nei supporti a **velocità lineare costante**, la densità di bit per traccia è uniforme: quindi tanto più una traccia è lontana dal centro del disco, tanto più sarà lunga e conterrà più settori; mentre nelle zone più interne del disco il numero di settori per traccia diminuisce e quando le testine si spostano in quest'area l'unità aumenta la velocità di rotazione per mantenere costante la quantità di dati che fluiscono sotto le stesse (questo metodo è utilizzato dalle unità ottiche come CD e DVD). Alternativamente la velocità di rotazione del disco può rimanere costante, mentre la densità di bit decresce spostandosi verso l'esterno per mantenere costante la quantità di dati che scorre sotto le testine. Questo metodo, noto come **velocità angolare costante**, è utilizzato nelle unità a disco magnetico.

## CONNESSIONE DEI DISCHI

I calcolatori accedono alla memoria secondaria in due modi: nei sistemi di piccole dimensioni il modo più comune è tramite le porte di I/O, oppure in modo remoto per mezzo di un file system distribuito.

## Memoria secondaria connessa alla macchina

Alla memoria secondaria connessa alla macchina si accede tramite le porte locali di I/O. I comuni pc usano bus di I/O IDE o ATA.

Le stazioni di lavoro a fascia alta impiegano architetture SCSI o FC.

-L’architettura SCSI è un’architettura a bus il cui supporto fisico è di solito un cavo piatto con un gran numero di conduttori. Consente di avere sul bus fino a 16 dispositivi. Il protocollo permette di accedere fino a 8 unità logiche per ciascun dispositivo di memorizzazione.

-L’FC è un’architettura seriale ad alta velocità che può funzionare sia su fibra che su cavo con 4 conduttori di rame e ci si aspetta grande potenzialità da questa connessione.

## Memoria secondaria connessa alla rete

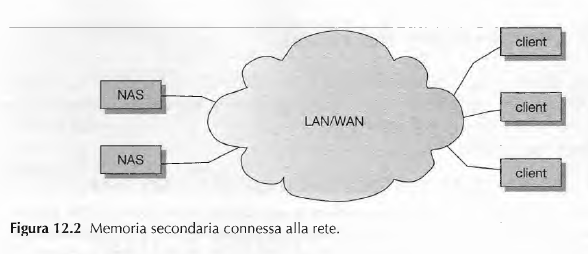
Ad un dispositivo di memoria connesso alla rete si accede in modo remoto grazie ad una rete di trasmissione dati. I client comunicano con la memoria connessa alla rete tramite interfacce come la **remote procedure call** (RPC) che funziona sul protocollo TCP/IP.

I client accedono alla memoria connessa alla rete tramite un’interfaccia RPC come NFS per Unix o CIFS per windows.

La memoria secondaria connessa alla rete fornisce un modo semplice per condividere spazio di memorizzazione a tutti i calcolatori di una lan con la stessa facilità di gestione dei nomi e degli accessi , caratteristica della memoria secondaria locale.

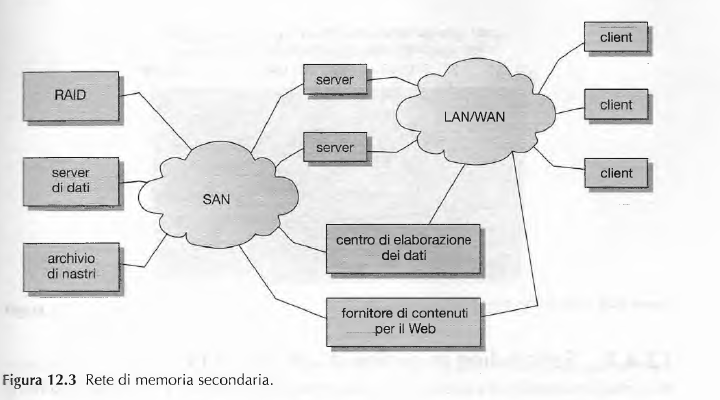
Tuttavia un sistema del genere ha prestazioni inferiori rispetto a memorie secondarie locali.

ISCSI è il più recente protocollo per la memoria secondaria connessa alla rete.



## Reti di memoria secondaria

Uno svantaggio dei sistemi di memoria secondaria connessa alla rete è che le operazioni di I/O sulla memoria secondaria impegnano banda della rete e quindi aumentano la latenza della comunicazione nella rete. Questo problema può essere particolarmente grave per sistemi client-server di grandi dimensioni. Per ovviare a questo inconveniente, molto sentito soprattutto in reti di grandi dimensioni, si utilizza una rete di memoria secondaria (**storage-area network**, o SAN), ovvero una rete privata che utilizza dei protocolli specifici per la memorizzazione tra i server e le unità di memoria secondaria,separata dalla LAN che collega i server ai client.



## 

## SCHEDULING DEL DISCO

Una delle responsabilità del sistema operativo è quella di fare un uso efficiente delle risorse fisiche, nel caso delle unità a disco far fronte a questa responsabilità significa garantire tempi d’accesso contenuti e ampiezze di banda elevate.

Il tempo di accesso dipende dal tempo di ricerca e dalla latenza di rotazione, mentre l'ampiezza di banda è il numero totale di byte trasferiti nel tempo intercorso dalla prima richiesta ed il completamento dell'ultimo trasferimento.

Quando un processo richiede di eseguire un'operazione di I/O lo fa tramite una chiamata di sistema.

La richiesta contiene diverse informazioni:

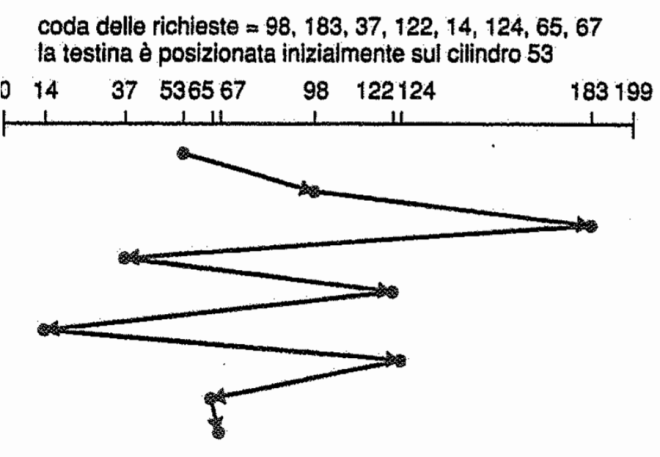
* Se l’operazione sia di immissione o emissione dati
* L’indirizzo del disco
* L’indirizzo di memoria al quale eseguire il trasferimento
* Il numero di byte da trasferire.

Se l’unità a disco desiderata e il controllore sono disponibili, la richiesta si può immediatamente soddisfare; altrimenti le nuove richieste si aggiungono alla coda di richieste inevase relativa a quell’unità. La coda relativa a un’unità a disco in un sistema con multiprogrammazione può spesso essere piuttosto lunga, sicché il SO sceglie quale fra le richieste inevase conviene scrivere prima.

## 

## Scheduling in ordine d’arrivo FCFS

E' l'algoritmo di scheduling più semplice che si possa immaginare (e implementare). Serve le richieste in coda in ordine di arrivo, non ottimizza l'utilizzo del disco ma non soffre di **starvation**.



## 

## Scheduling per brevità SSTF

**Shortest Seek Time First** è un algoritmo di scheduling che privilegia le richieste che fanno riferimento a zone vicine alla posizione attuale della testina, ovvero quelle che mantengono basso il **seek time** (che è anche il fattore che influenza di più i tempi di accesso al disco).

Trattandosi di fatto di un tipo di algoritmo SJF, può pero presentare il fenomeno della **starvation**, o

attesa indefinita (ad esempio se arrivano una serie di richieste relative ad un'area del disco, mentre

ce n'è già una relativa ad un'area più distante). L'algoritmo SSTF, sebbene migliore rispetto a FCFS,

non è ancora ottimale.

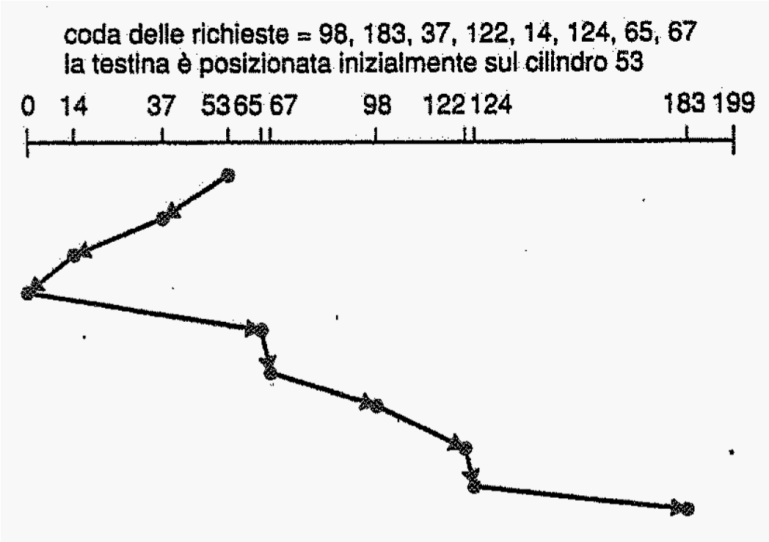
## 

## 

## Scheduling per scansione -SCAN

Se un disco utilizza uno scheduling di tipo SCAN, la testina si muove continuamente lungo il disco in una direzione, invertendo il verso solo quando raggiunge un'estremità dello stesso, e servendo le

richieste mano mano che incontra le posizioni interessate. Se una nuova richiesta arriva e la testina ha superato da poco tale posizione, essa deve attendere che la testina arrivi all’altra estremità e inverte la sua marcia. E' conosciuto anche come algoritmo dell’ascensore. Si noti che poche sono le richieste immediatamente vicine durante il cambio di marcia.

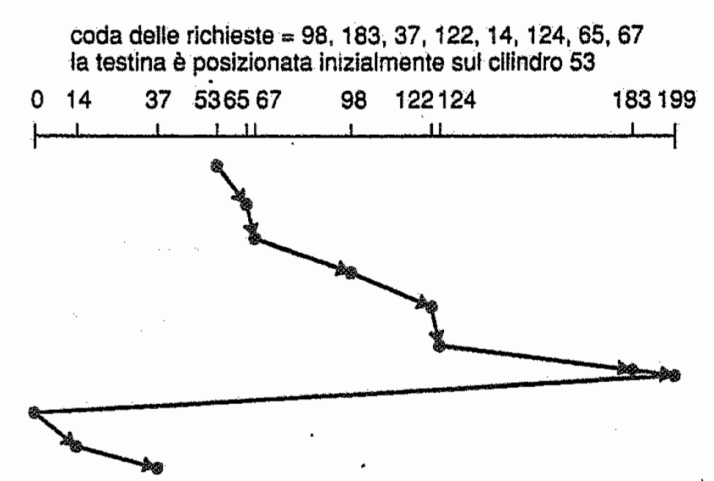


## 

## Scheduling per scansione circolare C-SCAN

Se si assume che le richieste avvengano in maniera uniforme nello spazio del disco, si può facilmente immaginare che, seguendo lo scheduling per scansione, quando la testina è arrivata alla fine del disco, il maggior numero di richieste pendenti sarà relativo all'estremo opposto.

Visto che queste sono anche le richieste in attesa da più tempo, appare ragionevole spostare la testina verso quest'area del disco. Questa è l'idea dell'**algoritmo C-SCAN**, che essenzialmente tratta il disco come

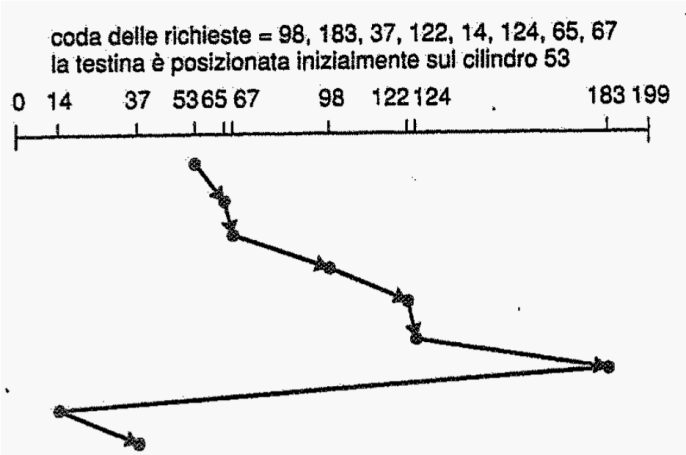
una lista circolare, riprendendo la scansione dalla prima posizione non appena raggiunta la fine del disco.

## 

## Scheduling look

Sia l'algoritmo SCAN che il C-SCAN, spostano la testina da un'estremità all'altra del disco, anche se non è detto che sia necessario raggiungere i settori più lontani.

Appare più conveniente invertire il senso dello spostamento delle testine quando non ci sono altre richieste da soddisfare nella direzione in cui si sta attualmente muovendo.

All'atto pratico, SCAN e C-SCAN vengono codificate in modo da comportarsi così, e questi due algoritmi prendono il nome di **LOOK** e **C-LOOK**, perchè “guardano” se ci sono altre richieste da soddisfare nella direzione attuale, prima di continuare a spostare la testina in quella direzione.

## 

## Scelta di un algoritmo di scheduling

Un algoritmo molto comune per lo scheduling del disco è sicuramente il SSTF, che migliora le prestazioni rispetto allo scheduling FCFS. Lo SCAN e il C-SCAN danno prestazioni migliori nei sistemi che sfruttano molto le unità disco. E' possibile stabilire, data una serie di richieste, quale sia l'algoritmo migliore per soddisfarle, ma il costo computazionale necessario sarebbe troppo elevato da portare un vero miglioramento rispetto ai risultati ottenibili con SSTF o LOOK.

Le richieste di I/O per le unità disco possono essere fortemente influenzate dal metodo utilizzato per

allocare i file: un programma che legge un file contiguo, genererà richieste raggruppate, con limitato movimento delle testine, mentre un'allocazione concatenata o indicizzata potrebbe richiedere letture sparse

per tutto il disco.

Anche la posizione delle directory è importante per l'utilizzo del disco, visto che per accedere ai file è necessario cercarne le informazioni relative alla loro posizione all'interno delle directory: se i dati di una directory sono nel primo cilindro, mentre il file da leggere effettivamente si trova nell'ultimo, la testina dovrà attraversare tutto il disco; viceversa se la directory si trova all'incirca al centro, la testina dovrà spostarsi al massimo per metà dell'ampiezza del disco. Anche l'uso della memoria centrale come memoria cache delle directory e dei blocchi indice può ridurre l'utilizzo del disco migliorandone l'efficienza.

Per questi motivi sarebbe auspicabile che la parte del sistema operativo che si occupi dello scheduling del disco costituisca un modulo a parte, consentendo di modificare l'algoritmo di scheduling in caso di necessità.

I controllori delle unità disco possono gestire autonomamente degli algoritmi di scheduling, ordinando le richieste del sistema operativo, ma questa non è sempre la scelta migliore: ad esempio potrebbe essere importante che dei fault di pagina che richiedano l'accesso al disco abbiano priorità rispetto alle operazioni di I/O dei processi, o che gli accessi in scrittura vengano eseguiti quanto prima quando la memoria cache sta per esaurire le pagine libere.

**GESTIONE DELL’UNITA’ A DISCO**

Il SO è anche responsabile di molti altri aspetti della gestione delle unità a disco.

## Formattazione del disco

Un disco magnetico nuovo è *tabula rasa*: un insieme di uno o più piatti privi di file system.

Per prima cosa deve essere diviso in settori e questa fase è detta **formattazione fisica** o di **basso livello**. La **formattazione di basso livello** riempie il disco di una struttura dati per ogni settore consistente di una intestazione , un'area per i dati e una coda.

L’intestazione e la coda contengono informazioni usate dal controllore del disco: ad esempio il numero del settore e un codice per la correzione degli errori EEC. Quando si effettua un’ordinaria operazione si aggiorna il valore EEC secondo il contenuto dell’area e i dati del settore.

Se risulta una discrepanza tra EEC e i dati letti nel settore significa che il settore potrebbe essere difettoso.

Per usare un disco come contenitore d’informazioni, il sistema operativo deve ancora registrare le proprie strutture dati nel disco, cosa che fa in due passi.

Il primo consiste nel suddividere il disco in uno o più gruppi di cilindri, detti **partizioni.** Il sistema operativo può trattare ogni gruppo come se fosse un’unita a disco a sè stante.

Il passo successivo alla suddivisione in partizioni è la **formattazione logica,** cioè la creazione di un file system: il sistema operativo registra nel disco le strutturedati iniziali relative al file system. Le strutture dati in questione possono includere descrizionidello spazio libero e dello spazio allocato **(FAT** o *inode)* e una directory iniziale vuota.

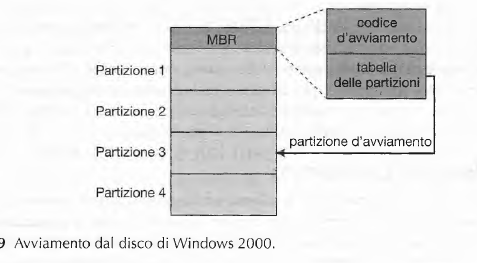
## 

## Blocco d’avviamento

Affinché un calcolatore possa entrare in funzione ad esempio quando viene acceso o riavviato è necessario che esegua un programma iniziale; di solito questo programma d’avviamento iniziale è piuttosto semplice. Esso inizializza un SO in tutti i suoi aspetti, dai registri della CPU all’avvio del SO. Per far ciò il programma d’avviamento trova il kernel del SO lo carica nella memoria e salta a un indirizzo iniziale per avviar l’esecuzione del SO. Per la maggior parte dei calcolatori il **blocco d’avviamento** viene memorizzato nella ROM che è una memoria di sola lettura. Poiché la ROM non è aggiornabile, nella ROM viene memorizzato un **caricatore d’avviamento**(bootstrap loader) il cui compito è quello di caricare da un disco il programma d’avviamento completo.

Il programma d’avviamento completo è registrato in una partizione del disco denominata **blocchi d’avviamento**, posta in una locazione fissa del disco chiamato **disco di sistema**.

Ad esempio l’esecuzione di windows : questo sistema colloca il proprio codice d’avviamento nel primo settore del disco chiamato MBR. Esso inoltre consente di suddividere il disco rigido in una o più partizioni; in una di loro detta **partizione d’avviamento** sono contenuti il SO e i driver di dispositivi. La procedura inizia con l’esecuzione del codice residente nella memoria ROM del sistema. Questa parte del codice guida il sistema a leggere il codice d’avviamento dall’MBR. Oltre al codice d’avviamento l’MBR contiene una tabella che elenca le partizioni del disco rigido e un flag che indica da quale partizione si debba avviare il sistema. Dopo aver identificato la partizione , il sistema legge il primo settore (**settore d'avviamento**) e svolge le restanti procedure d’avviamento.



## Blocchi difettosi

Le unità a disco sono strutturalmente portate ai malfunzionamenti perché sono costituite da parti mobili a bassa tolleranza. A volte si può verificare un guasto irreparabile, e l’unita a disco deve essere sostituita: le informazioni contenute nel disco dovranno essere recuperate da una copia di riserva mantenuta separatamente e trasferite nella nuova unità a disco.

Più di frequente, uno o più settori divengono malfunzionanti; in effetti, la maggior parte dei dischi messi in commercio contiene già **blocchi difettosi**. Essi sono trattati in diversi modi secondo il controllore e l’unità a disco presenti nel sistema.

Nel DOS bisognava chiamare una funzione del SO che permetteva di correggere o prendere soluzioni nel caso in cui uno di questi settori diventasse malfunzionante. Unità a disco più sofisticate come dischi SCSI hanno strategie di recupero dei **blocchi difettosi** molto raffinate.

Il controllore tiene una lista dei blocchi malfunzionanti dell’unità a disco che è inizializzata durante la formattazione fisica da parte del produttore e aggiornata man mano che si verificano nuovi malfunzionamenti. La formattazione mette anche a disposizione dei **settori di riserva** non visibili al SO nel caso in cui uno diventi difettoso. Un settore difettoso viene rimpiazzato da uno di questi settori. Questa strategia e chiamata **accantonamento di settori**.

Un tipico esempio è il seguente:

* Il SO tenta di leggere il blocco logico 87.
* Il controllore calcola l’ECC e scopre che il settore è difettoso , quindi segnala questo malfunzionamento al SO.
* La volta successiva che il sistema viene riavviato si esegue un comando speciale al fine di comunicare al controllore SCSI la necessità di sostituire il settore difettoso.
* Dopo di ciò ,ogni volta che il SO tenta di leggere il contenuto del blocco 87,il controllore traduce la richiesta nell’indirizzo del settore rimpiazzato.

Un’alternativa **all’accantonamento dei settori** è data da quei controllori capaci di sostituire i settori difettosi tramite la tecnica della **traslazione di settori.**(17->18 , .... , 202->203(che è di riserva)).

## GESTIONE DELL’AREA D’AVVICENDAMENTO

L’avvicendamento è stato introdotto dove abbiamo trattato lo spostamento di interi processi tra disco e memoria centrale. In quel contesto l’avvicendamento interviene quando l’ammontare della memoria fisica si abbassa fino al punto di raggiungere la soglia critica e i processi passano dalla memoria all’area di avvicendamento per liberare memoria. Nella pratica pochissimi SO realizzano l’avvicendamento nel modo descritto: essi infatti, combinano l’avvicendamento con tecniche di memoria virtuale per coinvolgere

nell’operazione solo alcune pagine e non necessariamente interi processi. Tant’è che alcuni sistemi considerano avvicendamento e paginazione termini intercambiabili.

La **gestione dell’area d’avvicendamento** è un altro compito di basso livello del sistema operativo.

La **memoria virtuale** usa lo spazio dei dischi come estensione della memoria centrale: poiché l’accesso alle unità a disco è molto più lento dell’accesso alla memoria centrale, l’uso di un’area d’avvicendamento riduce notevolmente le prestazioni del sistema. L’obiettivo principale nella progettazione e realizzazione di un’area di avvicendamento è di fornire la migliore produttività per il sistema della memoria virtuale.

## Uso dell’area d’avvicendamento

L’area d’avvicendamento è usata in modi diversi da sistemi operativi diversi in funzione degli algoritmi di gestione della memoria applicati. I sistemi che adottano l’**avvicendamento dei processi nella memoria** possono usare l’area di avvicendamento per mantener l’intera immagine del processo inclusi i segmenti dei dati e del codice; i sistemi a **paginazione** invece possono semplicemente memorizzarvi pagine non contenute nella memoria centrale. Lo spazio richiesto dall’area d’avvicendamento per un sistema può quindi variare secondo la quantità di memoria fisica, la quantità di memoria virtuale che esso deve sostenere, e il modo in cui quest’ultima è usata.

Un sistema che esaurisca l’area d’avvicendamento potrebbe essere costretto a terminare forzatamente i processi o ad arrestarsi completamente.

## Collocazione dell’area d’avvicendamento

Le possibili collocazioni per un’area d’avvicendamento sono due: all’interno del normale file system o in una partizione del disco a sè stante.

Se l’area d’avvicendamento è semplicemente un grande file all’interno del file system, si possono usare le ordinarie funzioni dei file system per crearla, assegnargli un nome e allocare spazio per essa. Questo criterio sebbene sia semplice da realizzare è inefficiente: l’attraversamento della struttura delle directory e l’uso delle strutture dati per l’allocazione dello spazio nei dischi richiede tempo. La frammentazione esterna può aumentare molto i tempi di avvicendamento causando ricerche multiple durante la scrittura e lettura dell’immagine del processo.

Le prestazioni si possono migliorare impiegando la memoria fisica come cache per le informazioni relative alla posizione dei blocchi e anche usando strutture speciali per l’allocazione in blocchi fisicamente contigui del file d’avvicendamento, ma il costo dovuto all’attraversamento del file system e delle sue strutture dati permane.

In alternativa all’area di avvicendamento si può creare un’apposta partizione del disco non formattata: in essa non è presente alcuna struttura relativa al file system e directory, ma si usa un particolare gestore dell’area d’avvicendamento per allocare e rimuovere i blocchi. L‘area di avvicendamento è inizializzata all’avvio, la frammentazione ha vita breve. Questo metodo assegna una dimensione fissa all’area d’avvicendamento al momento della creazione delle partizioni del disco e l’aumento delle dimensioni dell’area d’avvicendamento deve quindi passare attraverso il ripartizionamento del disco o attraverso la creazione di un’altra area d’avvicendamento in qualche altra unità a disco del sistema.

Alcuni SO non adottano una strategia rigida e possono costruire aree d’avvicendamento sia su partizioni specifiche sia all’interno del file system.

## STRUTTURE RAID

L’evoluzione tecnologica ha reso le unità a disco progressivamente più piccole e meno costose. Oggi è possibile equipaggiare sistemi con più dischi senza spendere somme esorbitanti.

Ci sono varie tecniche per l’organizzazione dei dischi, note col nome comune di **batterie ridondanti di dischi(RAID).**

Attraverso la configurazione RAID è possibile da un lato velocizzare l’accesso e da un altro effettuare copie di dischi qualora uno potrebbe rompersi.

Nel passato, strutture RAID composte da piccoli dischi economici erano viste come un’alternativa economicamente vantaggiosa rispetto a costosi dischi di grande capacità; oggi, le strutture RAID s’impiegano per la loro maggiore affidabilità e velocità di trasferimento dei dati, piuttosto che per ragioni economiche.

## 

## Miglioramenti dell’affidabilità tramite ridondanza

La possibilità che un disco in un insieme di *n* dischi si guasti è molto più alta della possibilità che uno specifico disco isolato presenti un guasto.

La soluzione dell’affidabilità sta nell’introdurre una ridondanza e quindi in caso di guasto basta sostituire il disco.

Il metodo più semplice (ma anche il più costoso) di introduzione di ridondanza è quello della **copiatura speculare (***mirroring* o *shadowing***);** ogni disco logico consiste di due dischi fisici e ogni scrittura si effettua in entrambi i dischi. Se uno dei dischi si guasta, i dati si possono leggere dall’altro. I dati si perdono solo se il secondo disco si guasta prima della sostituzione del disco già guasto.

Tuttavia, anche impiegando la copiatura speculare dei dischi, se si sta svolgendo un’operazione di scrittura nello stesso blocco in entrambi i dischi e si verifica una mancanza di tensione prima che sia completata la scrittura dell’intero blocco, i due blocchi possono ritrovarsi in uno stato incoerente. Una soluzione prevede la scrittura di una delle due copie e solo successivamente la scrittura della seconda, così che una delle due copie sia sempre coerente.

Un’altra soluzione è aggiungere una memoria cache non volatile **(NVRAM, *non-volatile RAM)*** alla batteria RAID, protetta dalla perdita di dati causata dalle cadute di tensione;se è dotata di forme di correzione d’errore come ECC o copiatura speculare i dati nella cachepossono essere considerati completi.

## 

## Miglioramento delle prestazioni tramite il parallelismo

L’accesso in parallelo di più dischi può portare grossi vantaggi. Con la **copiatura speculare** dei dischi , la frequenza con la quale si possono gestire le richieste di lettura raddoppia poiché ciascuna richiesta si può inviare indifferentemente a uno dei due dischi. Per migliorare la capacità di trasferimento i dati vengono distribuiti in sezione su più dischi. Questa forma nota come **sezionamento dei dati**, consiste nel distribuire i bit di ciascun byte su più dischi(in questo caso si parla di **sezionamento a livello dei bit**)facendo si che la quantità di dati letti o scritti sia 8 volte superiore. Questo però porta ad un grave svantaggio, nel caso in cui uno dei dischi si rompa, tutti i dati sono irrecuperabili.

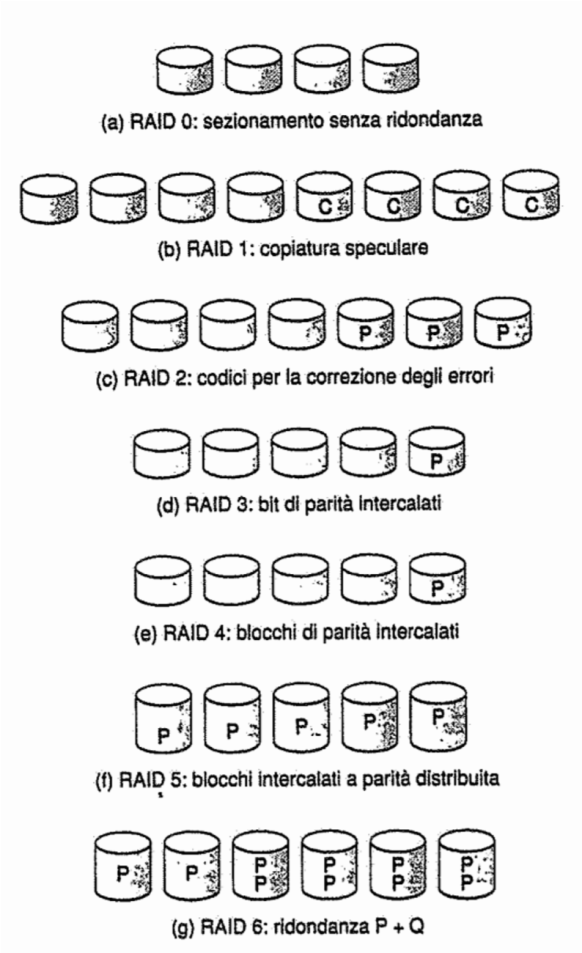
## 

## Livelli RAID

La tecnica di **copiatura speculare** offre un’alta affidabilità ma è costosa, la tecnica del sezionamento offre un’alta capacità di trasferimento ma non migliora l’affidabilità.

Sono stati proposti numerosi schemi per fornire ridondanza usando l’idea del sezionamento combinata con i bit di parità.

Questi schemi realizzano diversi compromessi tra costi e prestazioni e sono stati classificati in livelli chiamati **livelli RAID.**



* **Raid 0.** il livello 0 si riferisce a **batteria dei dischi con sezionamento a livello di blocchi** ma senza ridondanza ;
* **Raid 1.** il livello 1 si riferisce alla **copiatura speculare ;**
* **Raid 2.** noto anche come **organizzazione con codici per la correzione degli errori ECC**. Da molto tempo i dischi usano tecniche di riconoscimento degli errori basati sui bit di parità.

In un sistema di questo tipo, ogni byte di memoria ha associato un bit di parità che indica

se i bit con valore 1 nel byte sono un numero pari (parità = 0) oppure dispari (parità

= 1). Se si altera uno dei bit nel byte (un valore 1 diventa 0 o viceversa), la parità del byte cambia e quindi non concorda più con la parità memorizzata.

* **Raid 3.** Con il livello 3 o **organizzazione con bit di parità intercalati**, si migliora il raid 2 usando un unico bit di parità per individuare gli errori. Se uno dei settori è danneggiato si conosce esattamente di quale settore si tratta e calcolando la parità si conosce il suo valore.

È migliore rispetto al 2 quindi il 2 non viene proprio usato. È migliore dell’1 per 2 motivi: 1 perché si usa un solo disco per la parità anziché uno per ogni disco, 2 è che essendo la lettura e scrittura su byte distribuiti su più dischi la velocità aumenta di n volte.

* **Raid 4.** noto come **organizzazione con blocchi di parità intercalati**, s’impiega il sezionamento a livello dei blocchi come nel raid 0 e inoltre si tiene un blocco di parità in un disco separato. Se un dei dischi si guasta, il blocco di parità serve per recuperare il contenuto.
* **Raid 5.** noto come **organizzazione con blocchi intercalati a parità distribuita**, differisce dal 4 per il fatto che invece di memorizzare i dati in n dischi e la parità in un disco separati, i dati e le informazioni di parità sono distribuite tra gli n+1 dischi. Ogni blocco memorizza la parità e gli altri i dati. È il più comune sistema raid.
* **Raid 6.** Il livello 6 noto anche come **schema di ridondanza P+Q** è molto simile al 5 ma memorizza ulteriori informazioni ridondanti per potere gestire guasti contemporanei di più dischi. Invece della parità si usano **codici di Reed-Solomon**.

**-Raid di livello 0+1.** Il livello 0 + 1 consiste in una combinazione dei livelli RAID 0 e 1.

Il livello 0 fornisce le prestazioni, mentre il livello 1 l’affidabilità. Di solito, questo

schema porta a prestazioni migliori rispetto al livello 5 e si usa prevalentemente negli

ambienti in cui sono importanti sia le prestazioni sia l’affidabilità. Sfortunatamente,

questo schema richiede, come il RAID di livello 1, un raddoppio del numero di dischi

necessario per memorizzare i dati, quindi è anche più costoso del RAID di livello 5.

Nel RAID di livello 0 + 1, si sezionano i dati presenti in un insieme di dischi e si duplica

ogni sezione con la tecnica della copiatura speculare.

Un altro metodo che sta diventando disponibile commercialmente e il RAID di livello

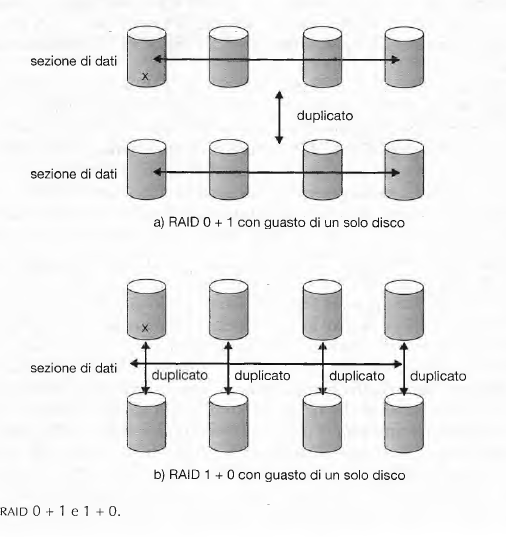
1 + 0, in cui si fa prima la copiatura speculare dei dischi a coppie, e poi il sezionamento

su queste coppie. Questo schema RAID ha alcuni vantaggi teorici rispetto al

RAID 0 + 1. Ad esempio, se si guasta un singolo disco nel RAID 0+1, l’intera sezione

di dati diventa inaccessibile, lasciando disponibile solo l’altra sezione. Con un guasto

nel RAID 1 + 0, il singolo disco diventa inaccessibile, ma il suo duplicato è ancora disponibile, come tutti gli altri dischi

****

Il raid può essere implementato sia a livello software che hardware. Un’altra caratteristica del raid è la previsione di **dischi di scorta** che possono sostituire immediatamente dischi rotti.

**STRUTTURE PER LA MEMORIZZAZIONE TERZIARIA**

## Dispositivi per la memorizzazione terziaria

La memoria terziaria consiste di mezzi rimovibili di cui dischetti, cd rom e dvd

**Dischi rimovibili**

I **dischi rimovibili** sono un tipo di memoria terziaria. I dischi magnetici rimovibili possono funzionare a una velocità quasi pari a quella di un’unità a disco,anche se il rischio che la loro superficie sia danneggiata da graffi è maggiore.

I dischi **magneto-ottici** sono un altro tipo di dischi rimovibili: anche in questo caso i dati sono registrati un disco ricoperto di materiale magnetico,ma la tecnologia impiegata per la registrazione è molto diversa da quella dei dischi magnetici .

**Nastri**

I **nastri magnetici** sono un altro tipo di memorizzazione terziaria. In generale un nastro può contenere più dati di un disco ottico o magnetico rimovibile. Le unità a nastro e quelle a disco hanno velocità di trasferimento simili ma poiché richiedono operazioni di avanzamento rapido o di riavvolgimento che possono durare anche alcuni minuti , l’accesso è molto più lento per un nastro. Grandi stazioni di registrazione a nastri usano tipicamente meccanismi automatici che riducono i costi totali della registrazione dei dati. Un archivio automatizzato realizza un tipo di memorizzazione talvolta detto quasi in linea, perché si situa fra le alte prestazioni della memorizzazione in linea nei dischi magnetici e il basso costo di una memorizzazione non in linea in nastri archiviati in qualche deposito.

**Tecnologie future**

Memorizzazione olegrafica, adopera la luce laser per memorizzare ologrammi su mezzi speciali.