Lezione 21 [02/12/22]

Capitolo 10: Memoria secondaria

- Struttura dei dispositivi di memorizzazione
- Struttura dei dischi
- Connessione dei dischi
- Scheduling del disco
- Gestione dell’unità a disco
- Gestione dell’area di avvicendamento
- Strutture RAID
- Realizzazione della memoria stabile

Struttura dei dischi

@redyz13 e @rosacarota

I moderni dischi dal punto di vista dell’indirizzamento si considerano come un grande vettore monodimensionale di **blocchi logici**, dove un blocco logico è la minima unità di trasferimento

La dimensione di un blocco logico è di solito 512 byte, ma alcuni dischi si possono **formattare a basso livello** allo scopo di ottenere una diversa dimensione dei blocchi logici (ad es 1024 byte)

Il vettore monodimensionale di blocchi logici corrisponde in modo sequenziale ai settori del disco:

- Il settore 0 è il primo settore della prima traccia sul cilindro più esterno

- La corrispondenza prosegue ordinatamente lungo la prima traccia, quindi lungo le rimanenti tracce del primo cilindro, e così via, di cilindro in cilindro, dall’esterno verso l’interno

I dischi girano secondo due tipi di velocità:

- **Velocità lineare costante**
  - La densità dei bit per traccia risulta uniforme. Più è lontana dal centro, tanto più è grande la traccia, tanto maggiore è il numero di settori che può contenere. L’unità aumenta la sua velocità di rotazione man mano che va verso il centro per mantenere costante la quantità di dati che scorrono sotto le testine
- **Velocità angolare costante**
  - La velocità di rotazione dei dischi rimane costante e la densità dei bit decresce dalle tracce interne alle tracce esterne per mantenere costante la quantità di dati che scorre sotto le testine

@redyz13 e @rosacarota

Dischi magnetici

I **piatti** dei dischi magnetici hanno forma piana e rotonda come quella dei CD

Le due superfici sono ricoperte di materiale magnetico su cui si memorizzano le informazioni

Una **testina** di lettura e scrittura è sospesa su ciascuna superficie d’ogni piatto

Le testine 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 divise in **settori**

L’insieme delle tracce corrispondenti ad una posizione del braccio costituisce un **cilindro**

In un’unità a disco possono esservi migliaia di cilindri concentrici e ogni traccia può contenere centinaia di settori

Quando un disco è in funzione, un motore lo fa ruotare ad alta velocità, questa velocità viene espressa in termini di giri al minuto **(RPM)**

La velocità 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
- Il **tempo di posizionamento**, detto anche tempo d’accesso casuale che consiste in due componenti:
  - Il tempo necessario a spostare il braccio del disco in corrispondenza del cilindro desiderato, detto **tempo di ricerca (seek time)**
  - Tempo necessario affinché il settore desiderato si porti, tramite la rotazione del disco, sotto la testina, detto **latenza di rotazione**

Poiché le testine di un disco sono sospeso su un cuscino d’aria sottilissimo (dell’ordine dei micron), esiste il pericolo che la testina urti la superficie del disco

In tal caso la testina potrebbe danneggiare la superficie magnetica (nonostante sia presente un sottile stato protettivo)

Tale incidente, detto **urto della testina**, di solito non può essere riparato e comporta la sostituzione dell’intera unità a disco

Un disco può essere **rimovibile**

- Ciò permette che diversi dischi siano montati secondo le necessità

Esempi di dischi rimovibili sono:

- I CD e DVD
- I Blu-ray
- Dispositivi di memoria flash (che costituiscono una tipologia di drive a stato solido)

Un’unità di disco è connessa a un calcolatore attraverso un insieme di fili detto **bus di I/O**

Esistono diversi tipi di tale bus, tra i quali:

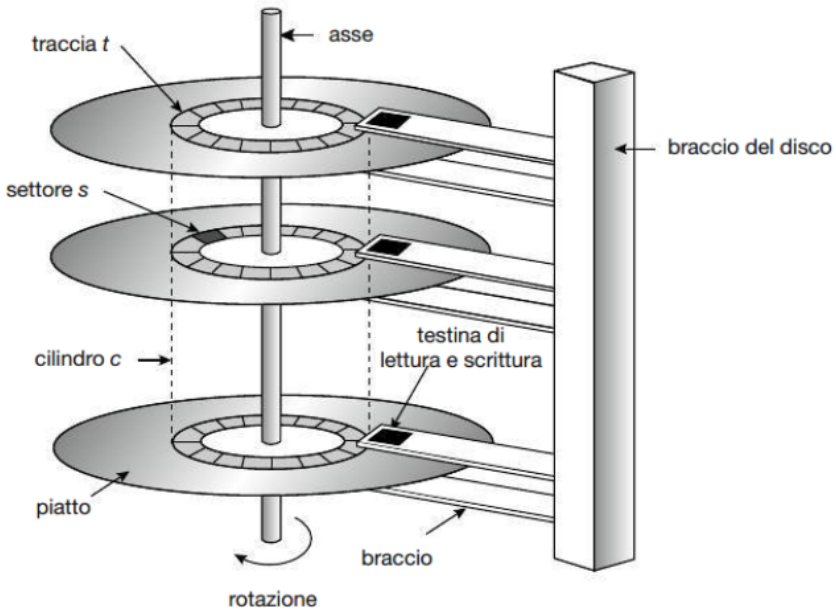
- ATA (Advanced technology attachment)
- SATA (Serial ATA)
- eSATA (External SATA)
- USB (Universal serial bus)
- FC (Fiber channel)

Il trasferimento dei dati in un bus è eseguito da processori dedicati detti **controllori**:

- I **controllori di macchina (host controller)** sono i controllori posti all’estremità del bus lato calcolatore
- I **controlli dei dischi (disk controller)** sono incorporati in ciascuna unità a disco

@ledy213 e @rosacarota

Schema di un disco a testine mobili:



Dischi a stato solido

Un **SSD** (disco a stato solido) è una memoria non volatile che viene utilizzata come un disco rigido

@redyz13 e @rosacarota

Ci sono molte varianti di questa tecnologia, dalle DRAM dotate di batteria (per permettere di mantenere lo stato in caso di caduta di tensione) alle tecnologie di memoria flash

Gli SSD hanno le stesse caratteristiche dei dischi rigidi tradizionali, ma possono essere:

- Più affidabili, perché non hanno parti in movimento
- Più veloci perché non hanno tempo di ricerca o di latenza

Inoltre consumano meno energia anche se hanno meno capacità rispetto ai dischi rigidi più grandi e possono avere durata di vita più breve

Alcuni SSD possono essere collegati direttamente al bus di sistema (PCI, per esempio) risultando di gran lunga più veloci

Nastri magnetici

I **nastri magnetici** sono stati i primi supporti di memorizzazione secondaria

Sono capaci di memorizzare in modo permanente un'enorme quantità di dati ma hanno un tempo d'accesso molto elevato (un migliaio di volte maggiore di quello dei dischi magnetici per via anche dell'accesso sequenziale dei nastri)

Vengono di solito utilizzati per la creazione di copie di backup dei dati o per registrare dati poco usati

Il nastro è avvolto in bobine e scorre su una testina di lettura e scrittura

Solitamente il posizionamento sul settore richiesto può richiedere alcuni minuti, anche se, una volta raggiunta la posizione desiderata, l'unità a nastro può leggere o scrivere informazioni a una velocità paragonabile a quella di un'unità a disco

@redyz13 e @rosacarota

Scheduling del disco

La velocità di un disco nel trasferire i dati è un fattore di notevole importanza per tutto il sistema di elaborazione

Il sistema operativo è responsabile per l'utilizzo efficiente dell'hardware

Per quanto riguarda il disco questo significa l'avere:

- Un **tempo di accesso** veloce
- Una grande **ampiezza di banda (bandwidth)**

    - Cioè il numero totale di byte trasferiti diviso il tempo totale intercorso tra la prima richiesta ed il completamento dell’ultimo trasferimento

Il sistema operativo può migliorare il tempo medio di servizio del disco pianificando le richieste di accesso al disco stesso attraverso un processo di scheduling

Il tempo di accesso da due componenti principali:

- **Tempo di ricerca:** il tempo necessario affinché il braccio dell’unità a disco sposti le testine al cilindro contenente il settore desiderato
- **Latenza di rotazione:** tempo aggiuntivo necessario perché il disco ruoti finché il settore desiderato si trovi sotto la testina

Tramite lo scheduling del disco si possono migliorare entrambe

Ogni volta che si devono compiere operazioni di I/O con un’unità a disco un processo effettua una system call

La richiesta contiene diverse informazioni:

@redyz13 e @rosacarota

- Se l’operazione è di input o di output
- L’indirizzo nel disco per il trasferimento
- L’indirizzo di memoria per il trasferimento
- Il numero di settori (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 richiesta inevase rispetto a quell’unità

La coda relativa ad un’unità a disco può essere spesso lunga, quindi il S.O. dovrà scegliere quale fra le richieste inevase converrà servire prima

Tale scelta è utilizzata per mezzo di uno dei vari algoritmi di scheduling presentati di seguito

Scheduling in ordine di arrivo (FCFS)

La forma più semplice di scheduling è l'algoritmo di servizio secondo l'ordine di arrivo (FCFS)

Nell'algoritmo di scheduling FCFS, la testina esegue le operazioni con lo stesso ordine di arrivo delle richieste

FCFS è un algoritmo semplice, ma non è ottimale in termini di velocità del servizio

È facile osservare infatti che le stesse richieste date in ordine diverso possono determinare tempi di servizio diversi

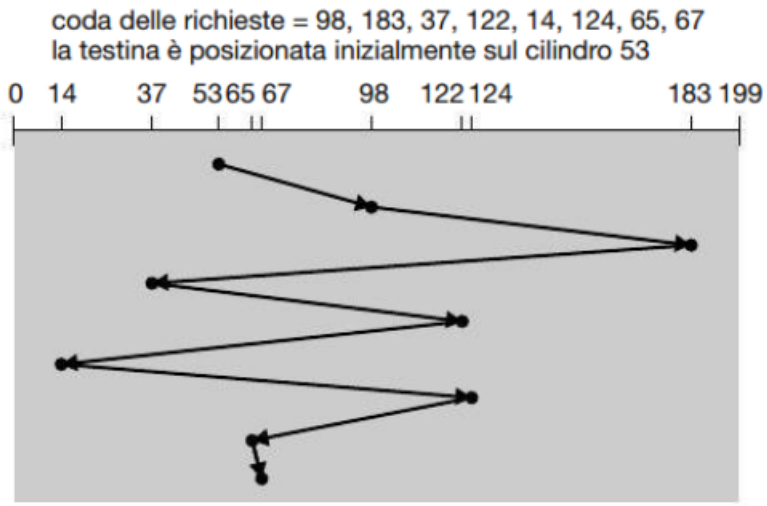
Tuttavia questo algoritmo viene utilizzato da quasi tutti i sistemi che hanno un carico basso, in quanto i pochi benefici ottenuti dall'utilizzo di altri algoritmi, su un sistema a basso carico, non riescono a compensare i costi sostenuti per realizzarli

Si consideri ad esempio una coda di richieste per unità a disco che dia una lista di cilindri sui quali individuare i blocchi richiesti nel seguente ordine: $98, 183, 37, 122, 14, 124, 65, 67$

La testina è inizialmente posta al cilindro 53

Scheduling FCFS:

Fredyz13 e Rosacarota



Scheduling per brevità (SSTF)

L’algoritmo di scheduling per brevità (Shortest Seek Time First), a partire dalla posizione corrente seleziona la richiesta con tempo di ricerca minimo

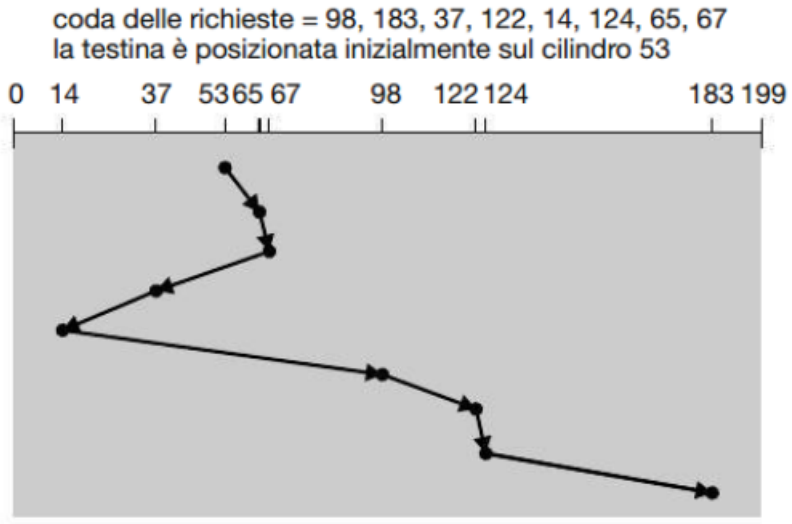
In pratica la testina si sposta sempre sulla traccia più vicina, prima di allontanarsi per servire un’altra richiesta

Lo scheduling SSTF è essenzialmente una forma di scheduling SJF (Shortest-job-first), ed al pari di questo può portare a situazioni di starvation

Infatti poiché le richieste possono arrivare in qualunque momento, la richiesta di una traccia lontana dalla posizione iniziale potrebbe rimanere per un tempo indefinito in attesa

Pur avendo prestazioni migliori del FCFS, l’algoritmo SSTF non è ottimale in termini di velocità del servizio

Scheduling SSTF:



Scheduling per scansione (SCAN)

Secondo l'algoritmo di scheduling per scansione il braccio dell'unità a disco parte da un estremo del disco e si sposta mantenendo la stessa direzione, servendo le richieste mentre attraversa i cilindri, fino a che non giunge all'altro estremo del disco

@fredyz13 e @rosacarota

A questo punto il braccio inverte la marcia e la procedura continua

Le testine attraversano continuamente il disco nelle due direzioni

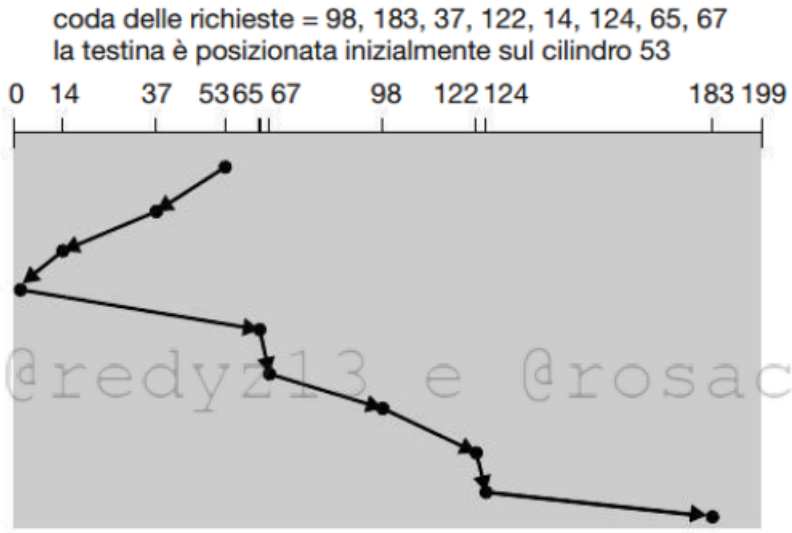
L'algoritmo **SCAN** viene talvolta chiamato **algoritmo dell'ascensore**, perché il braccio del disco si comporta come un ascensore che serve prima tutte le richieste in salita e poi tutte quelle in discesa

Se nella coda delle richieste di I/O giunge una richiesta di lettura per una traccia molto vicina alla testina (rispetto al suo verso corrente), questa viene servita quasi immediatamente

Invece la richiesta di una traccia che si trova dietro la testina deve attendere fino a che la stessa non arrivi all'estremità del disco, inverta la direzione di spostamento e ritorni

Prima di poter applicare lo scheduling SCAN, oltre la posizione corrente, occorre conoscere la direzione del movimento delle testine. Ad esempio, se lo spostamento è nella direzione del cilindro 0, l'unità servirà prima la richiesta 37

Scheduling SCAN:



@redyz13 e @rosacarota

Scheduling per scansione circolare (C-SCAN)

L'algoritmo di scheduling per scansione circolare (**Circular C-SCAN**) è una variante dell'algoritmo di scheduling SCAN progettato per fornire un tempo di attesa più uniforme

Anche il C-SCAN, come SCAN, sposta la testina da un estremo all'altro servendo le richieste lungo il percorso, ma quando la testina raggiunge l'altro estremo torna immediatamente all'inizio del disco stesso, senza servire richieste durante il ritorno

Ipotizzando una distribuzione uniforme per le richieste relative alle varie tracce, si consideri la densità di richieste che si presenta quando la testina raggiunge un'estremità e inverte la direzione di spostamento

A questo punto, dietro la testina sono presenti poche richieste in quanto queste tracce sono state servite di recente

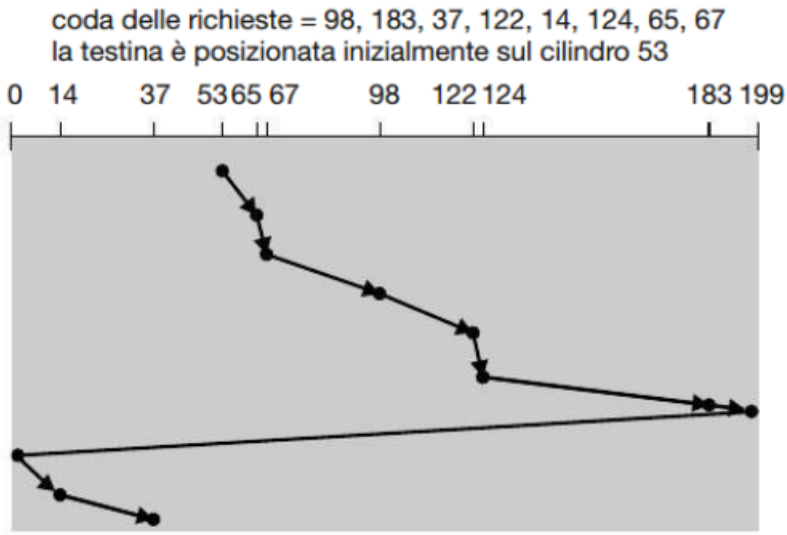
La maggiore densità di richieste è presente all'altra estremità del disco

Per ovviare a questa situazione quando la testina raggiunge l'estremità del disco, viene fatta tornare immediatamente all'inizio dello stesso senza servire le richieste che si trovano sul percorso di ritorno

Fondamentalmente, lo scheduling C-SCAN tratta il disco come una lista circolare, cioè come se il primo e l'ultimo cilindro fossero adiacenti

@redyz13 e @rosacarota

Scheduling C-SCAN:



Scheduling LOOK

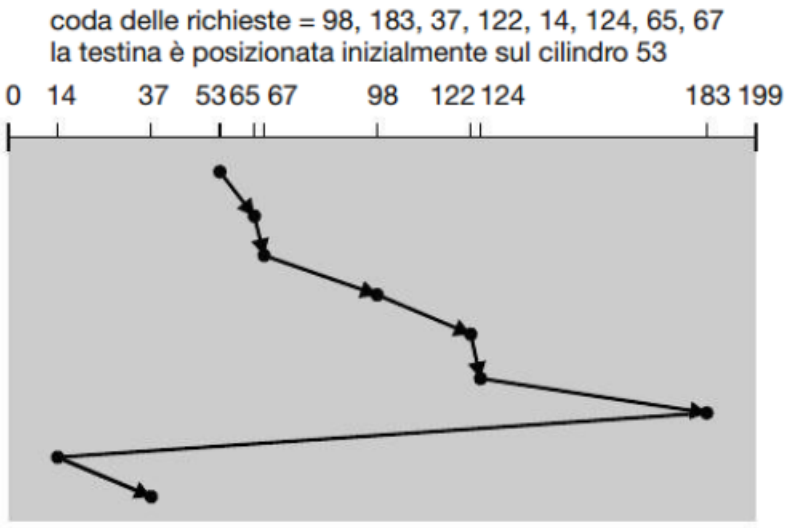
Gli algoritmi di scheduling LOOK e C-LOOK rappresentano un ulteriore passo avanti rispetto agli algoritmi SCAN e C-SCAN

Infatti:

@redyz13 e @rosacarota

- Gli algoritmi del tipo SCAN effettuano sempre delle scansioni complete del disco (cioè arrivano ad un estremo all'altro del disco)
- Gli algoritmi di tipo LOOK valutano ad ogni passo se vi sono altre richieste da servire in quella direzione, altrimenti viene immediatamente invertito il verso della testina

Scheduling C-LOOK:



Scelta di un algoritmo di scheduling

Uno degli algoritmi più utilizzati è l'SSTF, poiché aumenta le prestazioni rispetto all'FCFS

SCAN e C-SCAN danno migliori prestazioni in sistemi che sfruttano molto le unità a disco, poiché conducono con minore probabilità a situazioni di attesa indefinita

@eddy213 e @rosacarota

Definito l'insieme dei processi da schedulare, è possibile disegnare un algoritmo ad hoc ottimale, ma il risparmio che ne deriva non è sufficiente a compensare il calcolo necessario per ottenere l'ottimizzazione

È chiaro che la prestazione dipende fortemente dal numero e dai tipi di richieste

Se c'è di solito solo una richiesta in sospeso, tutti gli algoritmi si equivalgono (anche FCFS)

Gestione dell’unità a disco

Il processo di organizzazione della superficie del disco in tracce e settori che possano essere letti dal controllore è chiamato **formattazione di basso livello** o **formattazione fisica**, quasi tutti gli hard disk oggi vengono pre-formattati dalle case produttrici

La formattazione di basso livello riempie il disco con una specifica struttura di dati per ogni settore

In quasi tutti i sistemi, inclusi PC e MAC, i settori contengono in genere un’intestazione, un’area per i dati (in genera 512 bytes) ed una coda

L’intestazione e la coda contengono le informazioni usate dal controllore del disco

- Ad es. il numero di settore ed un **codice per la correzione degli errori (error-correcting code - ECC)**

L’ECC è un codice per la correzione dei dati: vuol dire che se solo alcuni bit di dati sono stati alterati, esso contiene sufficienti informazioni affinché il controllore possa identificare i bit alterati e ricalcolare il loro corretto valore

@redyz13 e @rosacarota

Il controllore riporta un **errore recuperabile** o **soft error.**
Il controllore esegue automaticamente l’elaborazione dell’ECC ogni volta che accede a un settore deln disco

Per utilizzare un disco, l’S.O. deve:

- **Partizionare** il disco in una o più partizioni, suddividendolo in gruppi di cilindri
- **Formattarlo logicamente,** cioè creare il file system (è un insieme di file + metodi di access ad esso)
    - La maggior parte dei file system accorpa i blocchi in gruppi, etti **cluster**. Quindi: l’I/O da disco procede per blocchi, ma l’I/O da file system procede per cluster

Alcuni S.O. danno la possibilità ad alcuni programmi di impiegare una partizione del disco come un array di blocchi logici sequenziale. questo array è detto alle volta **disco di basso livello (raw disc)** e l'I/O associato è detto **I/O di basso livello (raw I/O)**

Blocco di avviamento

Affinché un calcolatore possa entrare in funzione, è necessario che esegua un programma iniziale

Di solito il **programma di avviamento** (**bootstrap**) iniziale è molto semplice:

- inizializza il sistema in tutti i suoi aspetti, dai registri della CPU ai controllori dei dispositivi e al contenuto della memoria centrale, quindi avvia il SO

Per far ciò il programma di avviamento trova il kernel del sistema operativo sui dischi, lo carica nella memoriae e salta ad un indirizzo iniziale per avviare l'esecuzione del sistema operativo

In generale il programma di avviamento è memorizzato in **una memoria di sola lettura (ROM)**

@ready13 e @rosacarota

A causa di questo inconveniente molti sistemi memorizzano nella ROM un piccolo **caricatore di avviamento** (**bootstrap loader**), il cui solo scopo è caricare da disco il programma di avviamento completo, registrato in una partizione del **disco di avviamento** (o disco di sistema)

Il codice contenuto nella ROM istruisce i controllori del disco a affinché trasferisca il contenuto dei blocchi d'avviamento nella memoria

In MS-DOS il programma di avviamento è breve: un blocco di 512 byte

Blocchi difettosi

Le unità disco sono strutturalmente portate a malfunzionamenti perché costituite da parti mobili con basse tolleranze

Sebbene un ammontare straordinario di cura e sforzo da parte delle case produttrici vada nel costruire piatti per hard disk drives, non è economicamente fattibile fabbricare platters al 100% liberi da errori

Perciò, tutti i moderni drives adottano nel controller una strategia per il management degli errori per assicurare spazio libero da **blocchi difettosi (bad blocks)** (la maggior parte dei dischi messi in commercio già ne contiene)

Alla fine del processo di manifattura, l'intera superficie del disco è analizzata per evidenziare eventuali difetti e in questa fase il controllore del disco immagazzina una mappa delle loro ubicazioni

Accantonamento di settori - Quando il sistema operativo richiede che le informazioni debbano essere scritte in uno dei settori danneggiati, il controllore del disco trasparentemente lo mappa ad uno dei settori di riserva inutilizzati (creati tramite la prima formattazione fisica del produttore): i settori danneggiati sono sostituirti logicamente con uno dei settori di riserva inutilizzati

freddy213 e rosacarota

in parole povere, nel caso di indirizzi di blocchi difettosi, il controllore fa combaciare quegli indirizzi a un indirizzo in una determinata area y, quindi una volta che ci fosse un riferimento ad un un indirizzo di un blocco difettoso si salva nell'indirizzo dell0area y

Il controllore del disco aggiorna continuamente la mappa dei blocchi difettosi, per evitare che su nuovi settori danneggiati vengano memorizzate informazioni

Un'alternativa all'accantonamento dei settori è data da quei controllori capace di sostituire i settori difettosi con la **traslazione dei settori (sector slipping):** una volta che si

identifica un settore difettoso si cercai l primo settore libero non difettoso e si trasla di uno tutti i settori fino a salvare nel settore successivo al difettoso

Un **errore non recuperabile, hard error,** è un tipo di errore che porta alla è perdita di dati

Gestione dell’area di avvicendamento

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

La memoria virtuale usa lo spazio dei dischi come estensione della memoria centrale

I sistemi che adottano l’avvicendamento dei processi nella memoria possono usare l’area di avvicendamento (area di swap) per mantenere l’intera immagine del processo, inclusi i segmenti dei dati e del codice. Lo spazio per l’area di avvicendamento dipende quindi a seconda della quantità di memoria fisica, la quantità di memoria virtuale e il modo in cui è usata

I sistemi a paginazione possono semplicemente memorizzarvi pagine non contenute nella memoria centrale

@redyz13 e @rosacarota

Vi solo due possibili collocazioni per uno spazio di swap:

- All’interno del file system, quindi si usano le funzioni del file system per crearla (inefficiente)
- Su una partizione indipendente del disco; è una **partizione del disco non formattata (raw partition).** Potrebbe esserci frammentazione interna, ma viene ignorata perché l’area è reinizzializzata all’avvio del sistema

Alcuni S.O. possono creare aree di avvicendamento in entrambe le parti

Strutture RAID

La presenza di più dischi, qualora si possano usare in parallelo, rende possibile l'aumento della frequenza con cui i dati si possono leggere o scrivere, e permette di migliorare l'affidabilità della memoria secondaria poiché diventa possibile memorizzare le informazioni in più dischi in modo ridondante

- In questo caso, un guasto a uno dei dischi non comporta la perdita di dati

Esistono tecniche per l'organizzazione dei dischi note con il nome comune di **batterie ridondanti di dischi** (Redundant Array of Indipendent Disks - **RAID**)

Il RAID è un metodo di organizzazione dei dischi in un array che appare al calcolatore come una singola unità di memorizzazione

In passato strutture RAID composte da piccoli dischi economici erano viste come un'alternativa economicamente vantaggiosa rispetto a costosi dischi di grande capacità

Oggi si impiegano invece per la loro maggiore affidabilità e velocità di trasferimento dei dati

Quindi la I in RAID viene attualmente letta *independent* anziché *inexpensive* com'era originariamente

@redy213 e @rosacarota

RAID: affidabilità tramite la ridondanza

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

Se si memorizzasse una sola copia dei dati, allora ogni guasto di un disco comporterebbe la perdita di una notevole quantità di dati

La soluzione al problema dell'affidabilità sta nell'introdurre una certa **ridondanza**, cioè nel memorizzare informazioni che non

sono normalmente necessarie ma che si possono usare in caso di guasto per ricostruire le informazioni perse

Il metodo più semplice (ma anche il più costoso) di introduzione di ridondanza è quello della duplicazione di ogni disco

Questo metodo è detto **copiatura speculare** (**mirroring**)

- Ogni disco logico consiste di due dischi fisici e ogni scrittura si effettua su entrambi i dischi
- Si ottiene un disco duplicato (detto **mirrored volume**)
- 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

I casi di caduta di alimentazione elettrica costituiscono un noto problema

Anche impiegando il mirroring, se si sta svolgendo un'operazione di scrittura nello stesso blocco in entrambi i dischi e si verifica una caduta di alimentazione prima che sia completata la scrittura dell'intero blocco, i due blocchi possono ritrovarsi in uno stato incoerente

@redyz13 e @rosacarota

- Una soluzione prevede la scrittura di una delle due copie e solo successivamente la scrittura della seconda
- Un'altra soluzione è aggiungere una memoria non volatile a stato solido **(NVRAM, non-volatile RAM)** alla batteria RAID, protetta dalla perdita di dati dalle cadute di alimentazione
    - Se è dotata di forme di correzione d'errore come **ECC** o **mirroring**, la scrittura dei dati nella cache può essere considerata completa anche in quei casi

RAID: prestazioni tramite il parallelismo

Con la copiatura speculare, se si può accedere in parallelo a più dischi, la frequenza con la quale si possono gestire le richieste di lettura raddoppia

La capacità di trasferimento di ciascuna lettura è la stessa di quella di un sistema a singolo disco, ma il numero di letture per unità di tempo raddoppia

Attraverso l'uso di più dischi è possibile anche migliorare la capacità di trasferimento distribuendo i dati in sezioni su più dischi

La forma più semplice di questa distribuzione **(data striping)**, consiste nel distribuire i bit di ciascun byte su più dischi; in questo caso si parla di **sezionamento** o **striping a livello dei bit**

Per esempio, se il sistema impiega un array di otto dischi, si scriverà il bit i di ciascun byte nel disco i

Lo striping non si deve realizzare necessariamente a livello dei bit di un byte: nello **striping a livello di blocco**, per esempio, i blocchi di un file si distribuiscono su più dischi

Con n dischi, il blocco i di un file si memorizza nel disco $(i \ mod \ n) + 1$

@redyz13 e @rosacarota

Sono possibili anche altri livelli di striping ma questi sono i più comuni

Questa tecnica permette la riduzione del tempo di risposta relativo ad accessi a grandi quantità di dati e l'aumento, tramite il bilanciamento del carico, del throughput per accessi multipli a piccole porzioni di dati (cioè accessi a pagine)

Livelli RAID

Riassumendo:

- La tecnica di mirroring offre un'alta affidabilità ma è costosa

- La tecnica di striping (sezionamento) offre un’alta capacità di trasferimento dei dati, 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 più **livelli RAID**

- **Raid 0:** array di dischi con **sezionamento al livello dei blocchi**, ma senza ridondanza
- **Raid 1: copiatura speculare** (si riferisce alla tecnica di mirroring), ovvero tutti i dati sono memorizzati in due copie su diversi dischi (per quattro dischi di dati saranno necessari quattro dischi di overhead)
- **Raid 2: organizzazione con codici per la correzione degli errori di memoria**
  - **Bit di parità:**
    Ogni byte di memoria può avere associato un bit di parità che indica se i bit con valore 1 sono in numero pari (parità = 0) o in numero 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
    - In questo modo s’identificano tutti gli errori di un singolo bit nel sistema di memoria
    - Gli schemi di correzione degli errori memorizzano più bit supplementari e possono ricostruire i dati nel caso di un singolo bit danneggiato
  - Ogni disco memorizza un bit (striping dei byte) e altri dischi memorizzano i bit di correzione. Se uno dei dischi si guasta, i bit rimanenti del byte e i bit di correzione a esso associati si possono leggere dagli altri dischi e usare per ricostruire i dati danneggiati

Appunti di Rosacarota

    - Si noti che sono necessari solo 3 dischi di overhead per quattro dischi di dati
- **Raid 3: organizzazione con bit di parità intercalati**
    - Siccome i controllori dei dischi possono rilevare se un settore è stato letto correttamente, un unico bit di parità si può usare sia per individuare gli errori che per correggerli
    - Se uno dei settori è danneggiato, per ogni bit del settore è possibile determinare il suo valore considerando tutti gli altri bit dell'ottetto memorizzati sugli altri dischi (calcolando la parità dei bit corrispondenti dai settori negli altri dischi)
    - Se la parità dei bit rimanenti è uguale a quella memorizzata, il bit mancante è 0 altrimenti è 1
    - Più vantaggioso rispetto al RAID 2 siccome ha bisogno di un solo disco supplementare
- **Raid 4: organizzazione con blocchi di parità intercalati**
    - Si impiega il sezionamento dei blocchi (come nel raid 0) e si tiene un blocco di parità in un disco separato
    - Se uno dei dischi si guasta, il blocco di parità viene utilizzato insieme agli altri blocchi corrispondenti per ripristinare il blocco perso
- **Raid 5: organizzazione con blocchi intercalati a parità distribuita**
    - Differisce dal livello 4 perché invece di memorizzare i dati in n dischi e la parità in un disco separato, i dati e la parità sono distribuiti tra gli $n+1$ dischi
    - Per ogni blocco, uno dei dischi memorizza la parità e gli altri dati
    - Considerando una batteria di cinque dischi, la parità per il blocco $m-esimo$ si memorizza nel disco $(m\ mod\ 5)+1$,

@redyz13 e @rosacarota

mentre i blocchi $m-esimi$ degli altri quattro dischi contengono i dati effettivi per quel blocco

    - Un blocco di parità non può contenere informazioni di parità per blocchi che risiedono nello stesso disco, poiché un guasto provocherebbe anche la perdita dell'informazione di parità e i dati non sarebbero ripristinabili

- **Raid 6: schema di ridondanza P + Q**
    - Molto simile al 5, ma memorizza ulteriori informazioni ridondanti per gestire guasti contemporanei di più dischi Invece della parità si utilizzano i **codici di Reed-Solomon**
- **Raid 0 + 1:**
    - È una combinazione dei livelli 0 e 1
    - Si sezionano a livello dei blocchi i dati in un insieme di dischi e poi si duplica (indipendentemente ed eventualmente in modo diverso) ogni sezione con la tecnica della copiatura speculare
- **Raid 1 + 0:**
    - Si fa prima la copiatura speculare dei dischi a coppie e poi il sezionamento di queste coppie

Unedy213 e Rosacarota

Teoricamente in 0 + 1 se si guasta un disco l'intera sezione dati diventa inaccessibile lasciando disponibile solo l'altra sezione, mentre in 1 + 0 il singolo disco diventa inaccessibile, ma il suo duplicato è ancora disponibile

Schematizzazione dei livelli RAID:

(a) RAID 0: sezionamento senza ridondanza

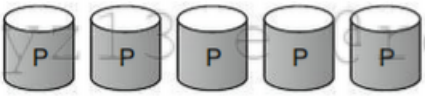
(b) RAID 1: mirroring

(c) RAID 2: codici per la correzione degli errori

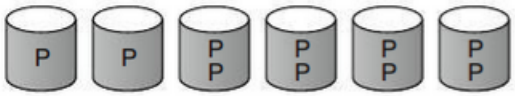
(d) RAID 3: bit di parità intercalati

(e) RAID 4: blocchi di parità intercalati

@redyz13e@tosacarota

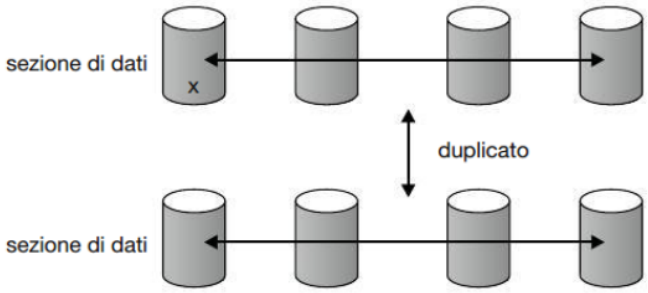


(f) RAID 5: blocchi intercalati a parità distribuita

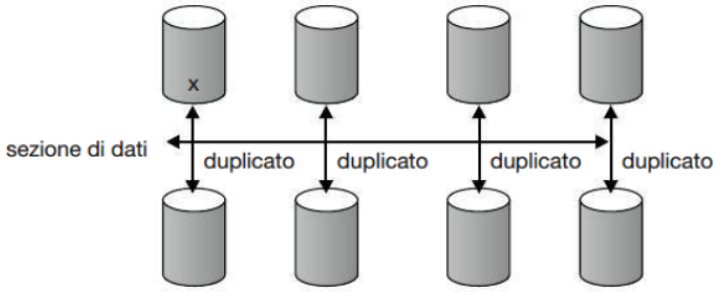


(g) RAID 6: ridondanza P + Q

RAID (0 + 1) e (1 + 0):



a) RAID 0 + 1 con guasto di un solo disco



b) RAID 1 + 0 con guasto di un solo disco

@redyz13 e @rosacarota

Scelta di un livello RAID

Se un disco si guasta il tempo per ricostruire i dati in esso memorizzati può essere rilevante e può variare secondo il livello RAID impiegato

La ricostruzione più semplice si ha al livello 1, poiché i dati possono essere semplicemente copiati dal duplicato del disco

Per altri livelli è necessario accedere a tutti i dischi della batteria

Il RAID di livello 0 si usa nelle applicazioni ad alte prestazioni in cui le perdite di dati non sono critiche

Il RAID di livello 1 si usa nelle applicazioni che richiedono un’alta affidabilità ed un rapido ripristino

I RAID 0 + 1 e 1 + 0 si usano dove entrambe prestazioni ed affidabilità sono importanti, per esempio per piccole basi di dati

Per la memorizzazione di grandi quantità di dati, in genere si preferisce impiegare il RAID di livello 5, non essendo il livello 6 sempre disponibile

I concetti relativi ai sistemi RAID sono stati generalizzati ad altri dispositivi

- Ad es. batterie di nastri o diffusione di dati in sistemi wireless

@redyz13 e @rosacarota