

Sistemi Operativi

Modulo 8:

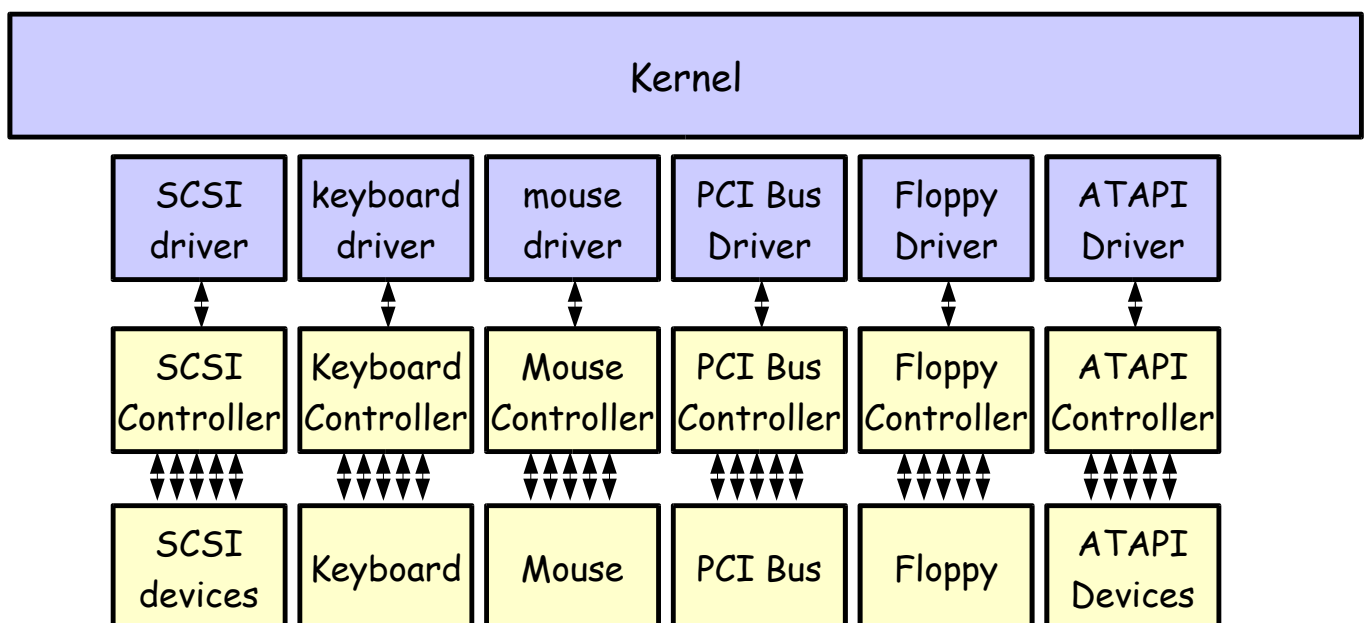
Gestione I/O, Memoria secondaria

Renzo Davoli
Alberto Montresor

Copyright © 2002-2005 Renzo Davoli, Alberto Montresor

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found at: <http://www.gnu.org/licenses/fdl.html#TOC1>

Interfacce I/O



Come classificare i sistemi di I/O

<i>Aspetto</i>	<i>Variazioni</i>	<i>Esempi</i>
Modalità di trasferimento	Caratteri Blocchi	Terminale Dischi
Modalità di accesso	Sequenziale Random	Modem CD-ROM
Trasferimento	Sincrono Asincrono	Nastri Mouse
Condivisione	Dedicato Condivisibile	Nastri Tastiera
Velocità	Pochi byte/s Gigabyte/s	Tastiere Schede di rete
Direzione di I/O	Sola lettura Sola scrittura Lettura/scrittura	Mouse Scheda video Dischi

Dispositivi a blocchi / caratteri

- **Interfaccia di comunicazione a blocchi**
 - i dati vengono letti/scritti a blocchi (tipicamente 512-1024 byte)
 - raw I/O
 - operazioni di read, write, seek per blocchi
 - accesso tramite file system
 - operazioni di read, write, seek su file
 - accesso tramite memory-mapped I/O
 - il contenuto di un file viene mappato in memoria
 - accesso tramite istruzioni di load/store del processore

Dispositivi a blocchi / caratteri

- **Interfaccia di comunicazione a caratteri**
 - i dati vengono letti/scritti un carattere alla volta
 - raw I/O
 - operazioni di get/put di un singolo carattere
 - bufferizzazione
 - lettura/scrittura di “una linea alla volta”

Progettazione del sistema di I/O

- **Tecniche di gestione dei dispositivi di I/O**
 - buffering
 - caching
 - spooling
 - I/O scheduling

Progettazione del sistema di I/O

- **Tre motivazioni per il buffering**

- per gestire una differenza di velocità tra il produttore e il consumatore di un certo flusso di dati
- per gestire la differenza di dimensioni nell'unità di trasferimento
- per implementare la “semantica di copia” delle operazioni di I/O

- **Caching**

- mantiene una copia in memoria primaria di informazioni che si trovano in memoria secondaria
- è differente dal buffering
 - nel buffer si trova l'unica *istanza* di un'informazione
 - la cache mantiene la *copia* di un'informazione

Progettazione del sistema di I/O

- **Spool**

- è un buffer che mantiene output per un dispositivo che non può accettare flussi di dati distinti
- ad esempio, stampanti

Caratteristiche dei dischi

• Struttura di un disco

- un disco è composto da un insieme di piatti, suddivisi in tracce, le quali sono suddivise in settori

• I dischi sono caratterizzati da tre parametri fondamentali

- r
 - la velocità di rotazione, espressa in *rpm* (*revolutions per minute*)
- T_s
 - il tempo di seek, ovvero il tempo medio necessario affinché la testina si sposti sulla traccia desiderata
- V_r
 - la velocità di trasferimento, espressa in byte al secondo

Caratteristiche dei dischi

• Il tempo di accesso

- è il tempo necessario per leggere un settore del disco, composto da tempo di seek, ritardo rotazionale e tempo di trasferimento

• Ritardo rotazionale

- il tempo medio necessario affinché il settore desiderato arrivi sotto la testina
- è uguale a $1 / 2r$

• Transfer time

- dipende dalla quantità di dati b da leggere (supponendo che siano contigui sulla stessa traccia)
- è uguale a b/V_r

Disk Scheduling

- **Gestione software dei dischi**

- il gestore del disco può avere numerose richieste pendenti, da parte dei vari processi presenti nel sistema
- il sistema sarà più efficiente se le richieste pendenti verranno evase seguendo un ordine che minimizza il numero di operazioni che richiedono molto tempo (e.g. seek)

- **Valori tipici**

- tempo di seek: 8-10 ms
- velocità rotazionale: 5400, 7200, 10000 rpm

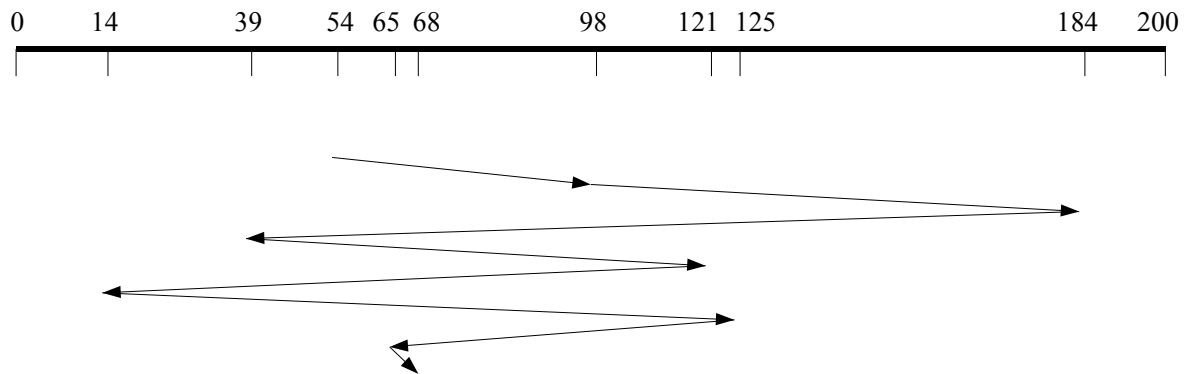
FCFS

- **First Come, First Served (altrimenti detta FIFO)**

- è una politica di gestione fair
- non minimizza il numero di seek
- non può mai generare starvation

FCFS - Esempio

- Coda delle richieste: 98, 184, 39, 121, 14, 125, 65, 68
- Posizione iniziale: 54



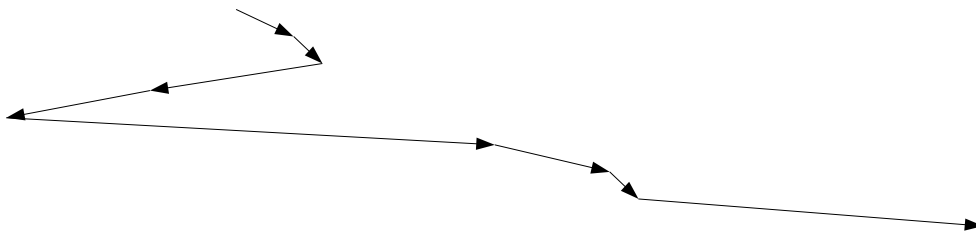
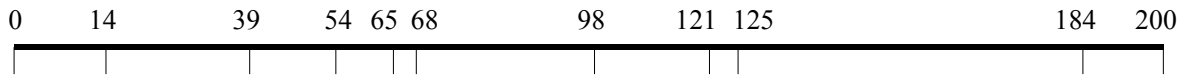
Lunghezza di seek totale: 639
media: 79.88

SSTF

- **Shortest Seek Time First**
 - seleziona la richieste che prevede il minor spostamento della testina dalla posizione corrente
 - nel caso di equidistanza, la direzione viene scelta casualmente
 - può provocare starvation

SSTF - Esempio

- **Coda delle richieste:** 98, 184, 39, 121, 14, 125, 65, 68
- **Posizione iniziale:** 54



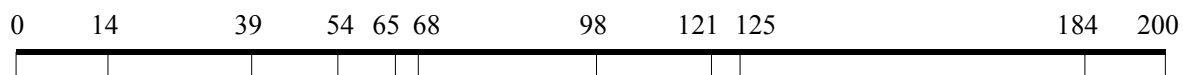
Lunghezza di seek totale: 238
media: 29.75

LOOK

- **Detto anche algoritmo dell'ascensore**
 - ad ogni istante, la testina è associata ad una direzione
 - la testina si sposta di richiesta in richiesta, seguendo la direzione scelta
 - quando si raggiunge l'ultima richiesta nella direzione scelta, la direzione viene invertita e si eseguono le richieste nella direzione opposta
- **Caratteristiche**
 - è efficiente
 - il tempo medio di accesso al disco non è omogeneo; sono privilegiate le tracce centrali
 - è esente da starvation (parzialmente, vedi prossimi lucidi)

LOOK - Esempio

- Coda delle richieste: **98, 184, 39, 121, 14, 125, 65, 68**
- Posizione iniziale: **54**



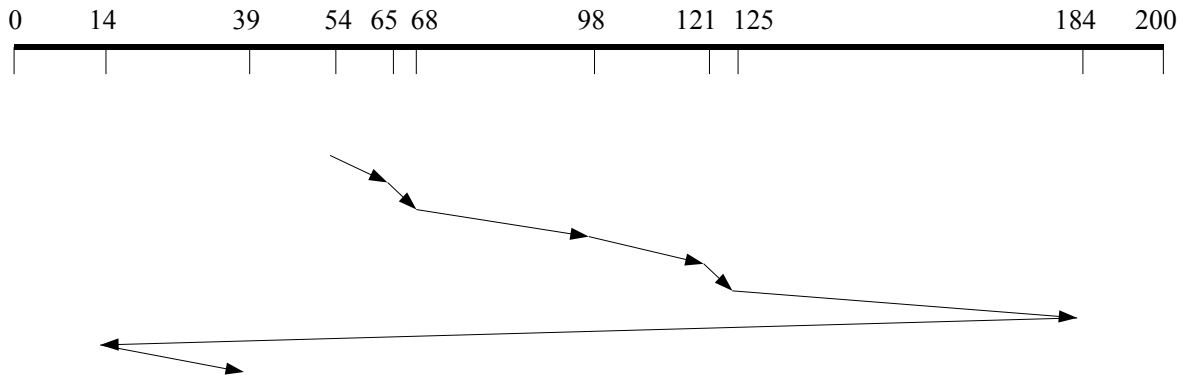
Lunghezza di seek totale: 210
media: 26.25

C-LOOK

- **C-LOOK**
 - ha lo stesso principio di funzionamento del metodo LOOK, ma la scansione del disco avviene in una sola direzione
 - quando si raggiunge l'ultima richiesta in una direzione, la testina si sposta direttamente alla prima richiesta,

C-LOOK - Esempio

- **Coda delle richieste:** 98, 184, 39, 121, 14, 125, 65, 68
- **Posizione iniziale:** 54



Lunghezza di seek totale: 325
media: 40.63

SSTF, LOOK e C-LOOK

- **Problema**
 - è possibile che il braccio della testina non si muova per un periodo considerevole di tempo
 - e.g., se un certo numero di processi continua a leggere sullo stesso cilindro
- **Soluzione**
 - la coda delle richieste può essere suddivisa in due sottocode separate
 - mentre il disk scheduler sta soddisfacendo le richieste di una coda, le richieste che arrivano vengono inserite nell'altra
 - quando tutte le richieste della prima coda sono state esaurite, si scambiano le due code

SCAN e C-SCAN

- **Nei testi**

- a volte questi algoritmi sono chiamati SCAN e C-SCAN
- altre volte, SCAN e C-SCAN sono due versioni distinte
 - si muovono tra gli estremi del disco (tracce più interne e esterne)
 - non partono/arrivano dalle richieste più interne o esterne

RAID

- **Problema**

- la velocità dei processori cresce secondo la legge di Moore, la velocità dei dispositivi di memoria secondaria molto più lentamente

- **Considerazioni**

- per aumentare la velocità di un componente, una delle possibilità è quella di utilizzare il parallelismo
- l'idea è quella di utilizzare un array di dischi indipendenti, che possano gestire più richieste di I/O in parallelo
- dobbiamo però garantire che i dati letti in parallelo risiedano su dischi indipendenti

RAID

- ♦ **Redundant Array of Independent Disks**
 - ♦ uno standard industriale per l'utilizzo di più dischi in parallelo
 - ♦ consiste di 7 schemi diversi (0-6) che rappresentano diverse architetture di distribuzione dei dati
- ♦ **Caratteristiche comuni ai sette schemi:**
 - ♦ un array di dischi visti dal s.o. come un singolo disco logico
 - ♦ i dati sono distribuiti fra i vari dischi dell'array
 - ♦ la capacità ridondante dei dischi può essere utilizzata per memorizzare informazioni di parità, che garantiscono il recovery dei dati in caso di guasti
- ♦ **Nota**
 - ♦ l'acronimo nell'articolo originale era *Redundant Array of Inexpensive Disks*

RAID

- ♦ **Considerazioni sui guasti**
 - ♦ l'utilizzo di più dischi aumenta le probabilità di guasto nel sistema
 - ♦ per compensare questa riduzione di affidabilità, RAID utilizza meccanismi di parità
- ♦ **Considerazioni sulle performance**
 - ♦ il data path che va dai dischi alla memoria (controller, bus, etc) deve essere in grado di sostenere le maggiori performance
 - ♦ il s.o. deve presentare al disco richieste che possano essere soddisfatte in modo efficiente
 - ♦ richieste di lettura di grandi quantità di dati sequenziali
 - ♦ gran numero di richieste indipendenti

RAID 0 (striping)

RAID Level 0

- non dovrebbe essere un membro "a tutti gli effetti" della famiglia RAID, perché non possiede meccanismi di ridondanza
- può essere utilizzato per applicazioni in cui l'affidabilità non è un grosso problema, ma lo sono la velocità e il basso costo

Descrizione

- i dati vengono distribuiti su più dischi
- vantaggi:
 - se due richieste di I/O riguardano blocchi indipendenti di dati, c'è la possibilità che i blocchi siano su dischi differenti
 - le due richieste possono essere servite in parallelo

RAID 0 (striping)

Striping

- il sistema RAID viene visto come un disco logico
- i dati nel disco logico vengono suddivisi in strip (e.g., settori, blocchi, oppure qualche altro multiplo)
- strip consecutivi sono distribuiti su dischi diversi, aumentando le performance della lettura dei dati sequenziali



RAID 0

• Performance di RAID 0

- per grandi trasferimenti di dati,
 - efficiente, in particolare se la quantità di dati richiesta è relativamente grande rispetto alla dimensione degli strip
- per un gran numero di richieste indipendenti
 - efficiente, in particolare se la quantità di dati richiesta è paragonabile alla dimensione degli strip

• Ridondanza

- nessuna

RAID 1 (mirroring)

• RAID level 1

- differisce dagli schemi 2-6 per come la ridondanza è gestita
- la ridondanza è ottenuto duplicando tutti i dati su due insiemi indipendenti di dischi
- come prima, il sistema è basato su striping, ma questa volta uno strip viene scritto su due dischi diversi
- il costo per unità di memorizzazione raddoppia



RAID 1 (mirroring)

• Performance di RAID 1

- una richiesta di lettura può essere servita da uno qualsiasi dei dischi che ospitano il dato
 - può essere scelto quello con tempo di seek minore
- una richiesta di scrittura può essere servita da uno qualsiasi dei dischi che ospitano il dato
 - dipende dal disco con tempo di seek maggiore

• Ridondanza di RAID 1

- il recovery è molto semplice;
 - se un disco si guasta, i dati sono accessibili dall'altro disco
 - è necessario sostituire il disco guasto e fare una copia del disco funzionante

RAID 2-3 (parallel access)

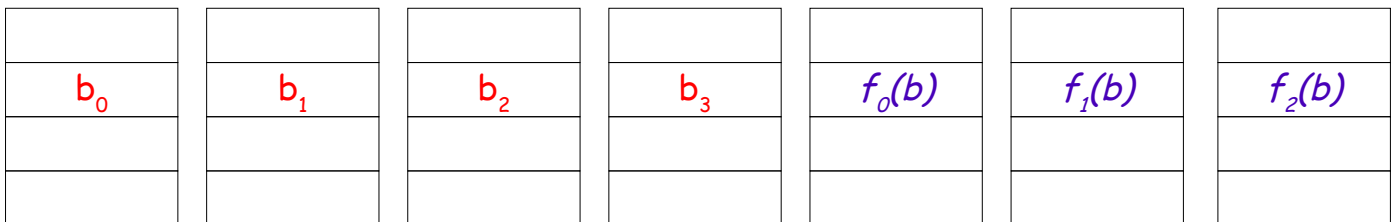
• RAID level 2-3

- *accesso parallelo*
 - tutti i dischi partecipano all'esecuzione di ogni richiesta di I/O
- i dischi sono sincronizzati in modo che le testine di lettura siano nella stessa posizione allo stesso istante
- si utilizza data striping, con gli strip molto piccoli (byte, word)
- i dischi vengono suddivisi fra *dischi dati* e *dischi parità*
 - un codice di correzione di errore o di parità viene calcolato a partire dai bit corrispondenti dei dischi dati
 - questo codice viene memorizzato nei dischi parità

RAID 2-3

RAID level 2

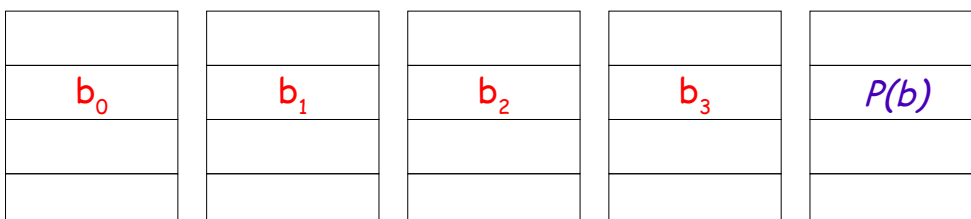
- il codice calcolato è basato sul codice di Hamming
- il numero di dischi di parità è dato dal logaritmo del numero di dischi di dati
- permette di correggere errori fino a un bit (e di rilevare errori fino a due bit)
- non molto utilizzato, perchè utile solo in ambienti con un alto numero di errori di disco



RAID 2-3

RAID level 3

- il codice calcolato è un semplice bit di parità
- è richiesto un solo disco di parità



RAID 2-3

• Ridondanza di RAID 3

- supponiamo di avere quattro dischi dati (numerati 0-3) e un disco di parità (numerato 4)
- la parità per il quarto disco è calcolata nel modo seguente
 - $b_4(i) = b_0(i) \oplus b_1(i) \oplus b_2(i) \oplus b_3(i)$
- nel caso di guasto di un disco (diciamo il disco 1), è possibile calcolare il valore corrispondente utilizzando il disco di parità
 - $b_1(i) = b_0(i) \oplus b_4(i) \oplus b_2(i) \oplus b_3(i)$

• In caso di guasto

- il sistema è in modalità "ridotta"
- bisogna sostituire il disco e rigenerare i dati mancanti

RAID 2-3

• Performance di RAID 2-3

- sono più adatti per sistemi che richiedono alte velocità di trasferimento dati
 - i dati sono "striped" in quantità molto piccole
 - il sistema può raggiungere alte velocità di trasferimento dati, leggendo/scrivendo su tutti i dischi contemporaneamente
- poiché ogni lettura coinvolge tutti i dischi, RAID 2-3 non sono adatti per sistemi con un grande numero di richieste di I/O indipendenti

RAID 4-6

RAID level 4-6

- *accesso indipendente*
 - ogni disco opera in modo indipendente, in modo che richieste indipendenti possano essere accedute in parallelo
- come in RAID 3, si utilizza un disco di parità

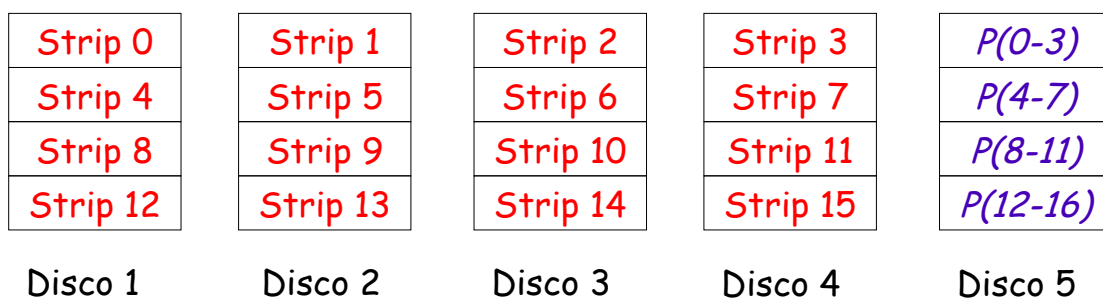
Performance

- sono più adatti per sistemi con un gran numero di richieste indipendenti
- un po' meno adatti per sistemi che richiedono il trasferimento di grandi quantità di dati

RAID 4

Descrizione

- si utilizza il meccanismo di data striping, con strip relativamente grandi
- strip di parità
 - viene calcolato uno strip di parità, a partire dagli strip di dati corrispondenti, calcolato bit-per-bit
 - lo strip di parità viene posto sul disco di parità



RAID 4

- **Letture (in assenza di guasti)**
 - si individua lo strip corrispondente e si effettua la lettura dello strip
- **Letture (in presenza di guasti)**
 - si individua lo strip corrispondente; se il disco corrispondente è guasto, si effettua la lettura di tutti gli strip rimasti e tramite il disco di parità si ottiene lo strip mancante
- **Scrittura (in assenza di guasti)**
 - quanti strip devono essere coinvolti?
 - a prima vista, si direbbe tutti (lo strip dati da scrivere, tutti gli altri strip da leggere, lo strip di parità in scrittura)

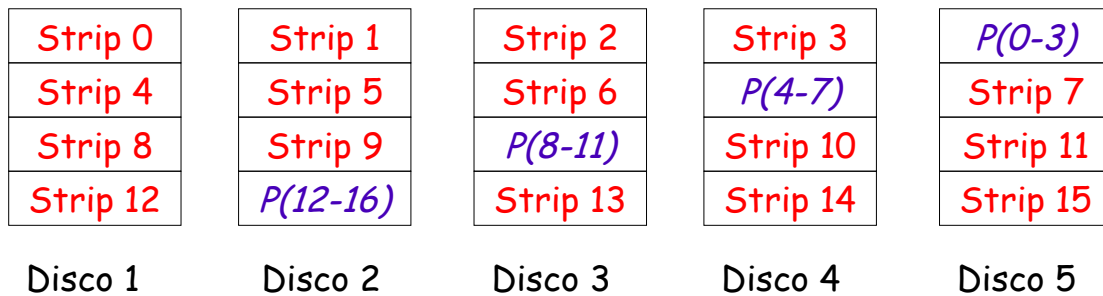
RAID 4

- **Scrittura (in assenza di guasti)**
 - in realtà, ne bastano 3
 - spiegazione
 - supponiamo che si voglia cambiare lo strip 1; il calcolo da effettuare è questo
$$S'_4(i) = S_0(i) \oplus S'_1(i) \oplus S_2(i) \oplus S_3(i)$$
dove $S'_1(i)$ è il nuovo valore dello strip da scrivere, e $S'_4(i)$ è il nuovo valore dello strip di parità da scrivere
 - possiamo scrivere
$$S'_4(i) = S_0(i) \oplus S'_1(i) \oplus S_1(i) \oplus S_1(i) \oplus S_2(i) \oplus S_3(i)$$
$$S'_4(i) = S_4(i) \oplus S'_1(i) \oplus S_1(i)$$

RAID 5

• Descrizione

- come RAID 4, ma i blocchi di parità sono sparsi fra i vari dischi
- il vantaggio è che non esiste un disco di parità che diventa un bottleneck



RAID 6

• Descrizione

- come RAID 5, ma si utilizzano due strip di parità invece di uno
- aumenta l'affidabilità (è necessario il guasto di tre dischi affinché i dati non siano utilizzabili)