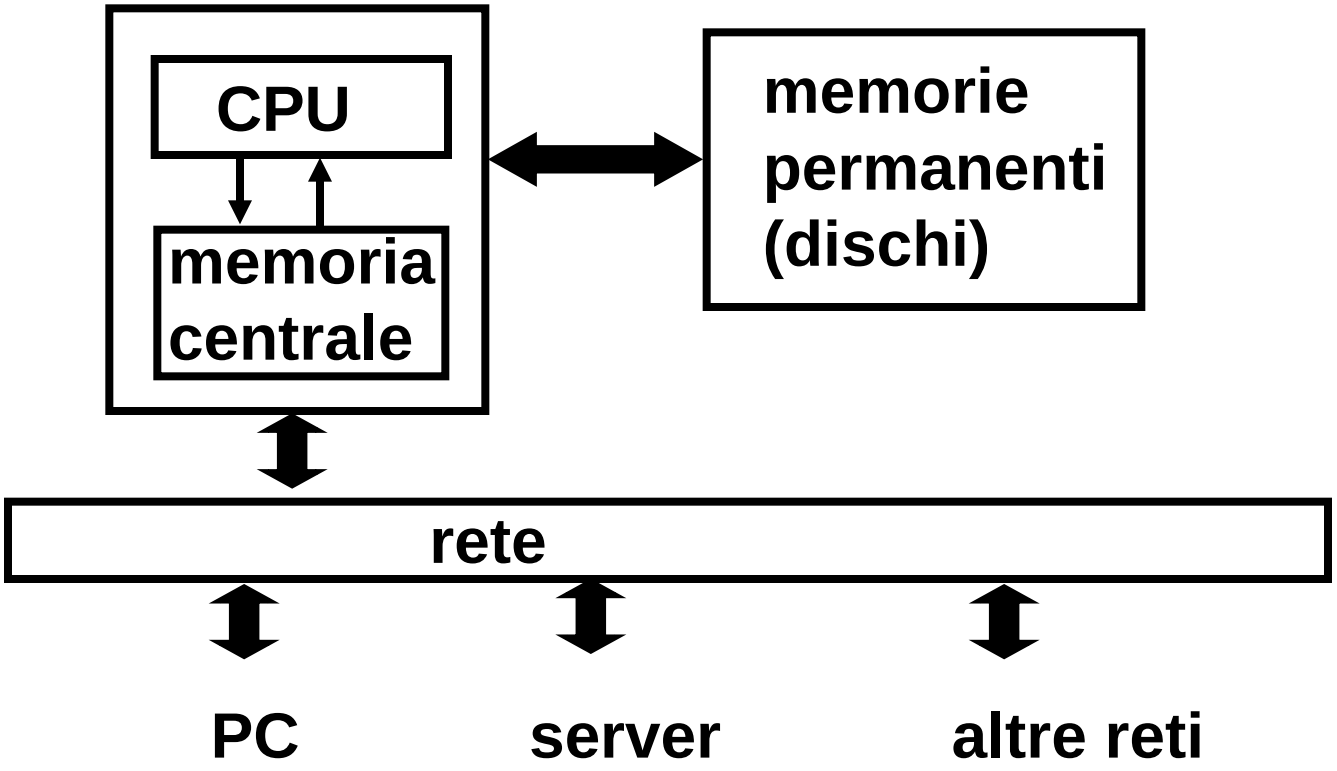
PARALLELISMO NELLE MEMORIE PERMANENTI

Struttura di un data server



Qualità di un data server

- **velocità della CPU**
- **capacità e velocità della memoria centrale (…o memoria principale)**
- **capacità e velocità delle memorie permanenti (...o memorie secondarie)**

  – **si tende ad enfatizzare le prime due mentre per noi la più importante è la terza!**

  perché condiziona la velocità del servizio nelle applicazioni gestionali

Qualità di un data server

- La qualità della CPU, a parità di tecnologia elettronica costruttiva, si misura in velocità del clock (es. in Ghz) e in numero di bit dei registri (32-64)
- le prestazioni generali tendono e migliorare di circa 1.5 volte ogni anno (negli ultimi anni)
- i costi sono in forte calo (a parità di prestazioni)

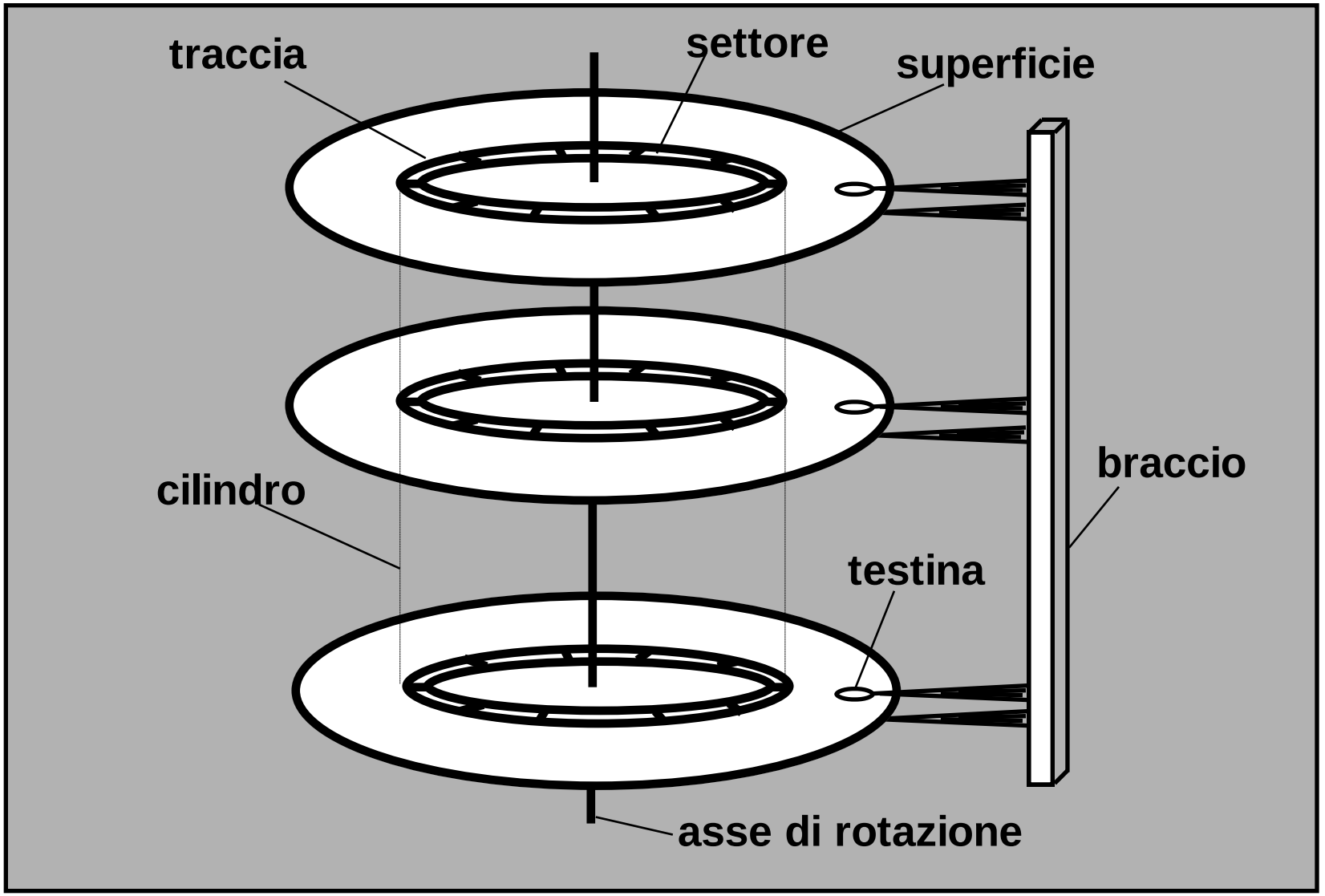
Qualità di un data server

- **secondo la legge di Amdahl:**

  speedup = 1 / [(1-f) +f/k]

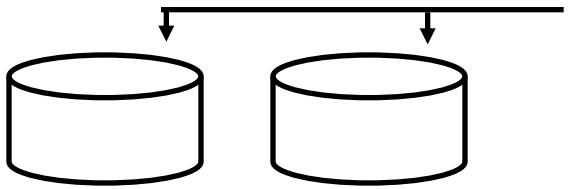
  [*f*= tempo usato dalla CPU e *k*= speedup CPU]
- **tenendo ferma la tecnologia delle memorie permanenti, se le operazioni di I/O incidono per un 10% (f=0.9), aumentando la velocità della CPU di un fattore k=10 si avrebbe un miglioramento delle prestazioni del server di un fattore 5; aumentando di un fattore 100 si avrebbe un miglioramento di solamente 10**
- **ciò giustifica lo sforzo dell'industria per adeguare le prestazioni delle memorie permanenti**

Struttura di un disco rigido



parallelismo e sicurezza

Disk Mirroring: dischi con dati replicati



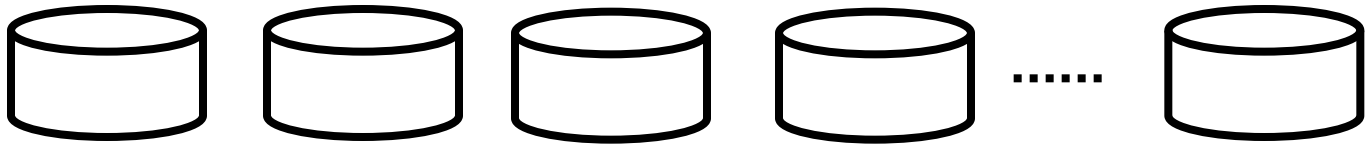
letture indipendenti
scritture su entrambi

- **aumentando il numero di dischi aumenta la probabilità di averne uno guasto**
- **diminuisce la probabilità di perdita dei dati dovuta al guasto contemporaneo**
- **ridondanza eccessiva è da evitare ma…**

la ridondanza è utile!

dischi RAID

Redundant Array of Inexpensive Disks



- **architettura che migliora le prestazioni e/o l'affidabilità del sistema di mem. permanente**
- **l'uso di N dischi consente di suddividere i dati in piccoli blocchi da scrivere e leggere in parallelo**
- **L'introduzione di informazioni ridondanti consentono la correzione di errori dovuti a guasti**

Parallelismo nei dischi

- **Il parallelismo è ottenuto grazie a due tecniche di base:**

STRIPING:
ciascun blocco è scritto contemporaneamente su tutti i dischi con un movimento sincrono delle testine (ogni blocco viene così distribuito byte per byte o addirittura bit per bit sui vari dischi)

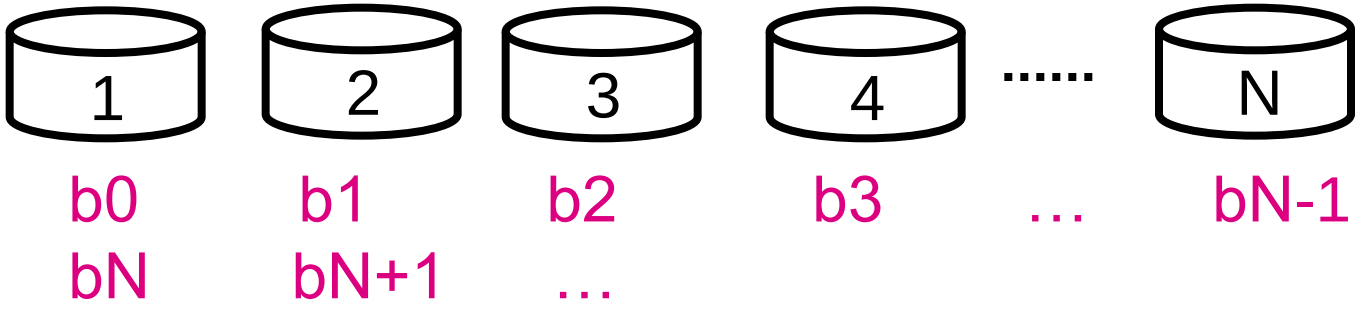
DECLUSTERING:
ciascun blocco è scritto intero su di un disco, blocchi successivi di uno stesso file sono scritti in parallelo ed in maniera indipendente sui vari dischi

Dischi RAID

- **servizio parallelo indipendente per letture brevi per più utenti (parallelismo inter-query) → declustering**
- **servizio parallelo per letture lunghe per lo stesso utente (parallelismo intra-query) → striping**

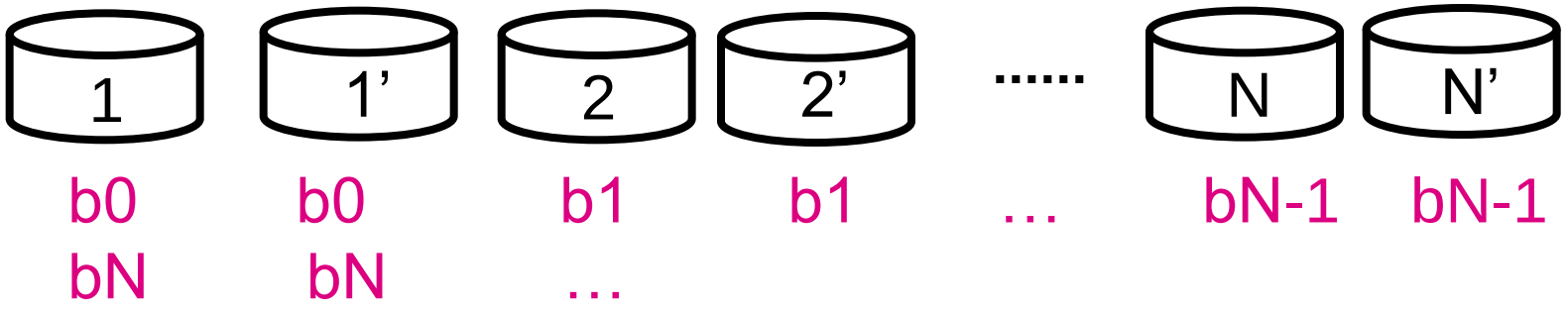
- **RAID 0: nessuna ridondanza**
- **RAID 1: mirroring – massima ridondanza**
- **RAID 2-3: bit di controllo o di parità su dischi dedicati; prevede unico movimento parallelo delle teste (striping)**
- **RAID 4: prevede un disco dedicato per l'informazione ridondante di parità**
- **RAID 5: l'informazione di parità viene distribuita ciclicamente sui vari dischi**

RAID 0



- **I blocchi di un file sono distribuiti ciclicamente sui vari dischi**
- **Con N dischi si ha un incremento di un fattore N delle prestazioni in lettura e scrittura grazie al parallelismo delle operazioni**
- **Non c'è ridondanza quindi non c'è tolleranza ai guasti (la rottura di un disco comporta la perdita di tutti i suoi dati)**

RAID 1



- **Ciascun disco ha un “gemello” (mirror) e ciascun blocco viene quindi scritto su due dischi**
- **Per ogni coppia di “gemelli” sono possibili una scrittura o due letture indipendenti in parallelo**
- **C’è massima ridondanza (tutti i dati sono duplicati) ed elevata tolleranza ai guasti**

Uso della parità

Concetto di parità:

- **dati 8 bit + un nono bit di parità calcolato come la somma modulo 2 dei bit di dati (0 se il numero di 1 nel byte è pari altrimenti è 1)**

  0 0 1 0 1 1 0 1 → 0

  1 0 1 1 0 1 1 0 → 1

- **se un bit si inverte la parità non torna quindi si sa che c'è un bit errato (però non si sa qual'è)**

- **nel RAID 4 i bit di parità stanno tutti sul nono disco**

Uso della parità (RAID 4)

- **Se un disco si guasta il controller se ne accorge e l'informazione viene ricostruita con la parità:**

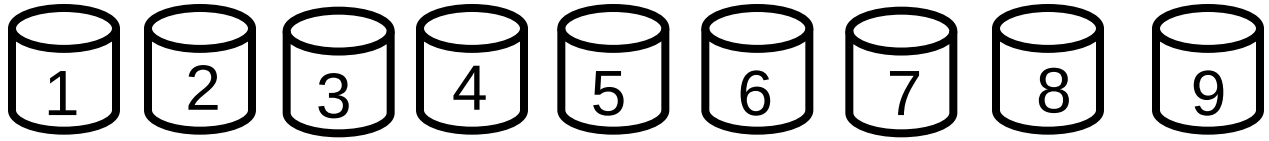
0 0 1 0 x 1 0 1 → 0 poiché la parità è 0 il bit era 1

1 0 0 1 x 1 1 0 → 1 poiché la parità è 1 il bit era 1

0 1 1 0 x 0 0 1 → 1 poiché la parità è 1 il bit era 0

Il RAID 4 ha un eccesso di letture e scritture sul nono disco

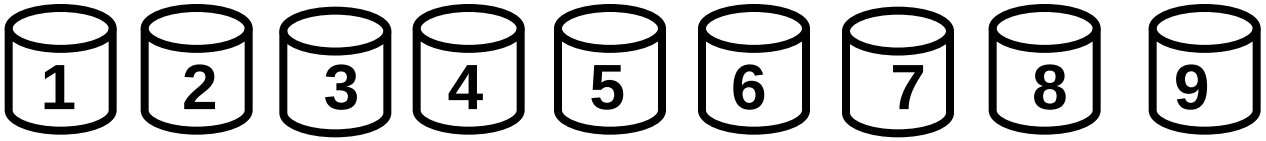
RAID 5



Il RAID 5 ha 1 solo disco in più ma parità e dati sono distribuiti ciclicamente su tutti: un blocco di dati viene memorizzato sui dischi da 1 a 8 e la parità sul 9, un secondo blocco dati viene memorizzato sui dischi da 9 a 7 e la parità sul 8 ….

Il RAID 6 ha 2 dischi in più ma l'informazione è codificata con il codice Reed Solomon e riesce a correggere due guasti

RAID 5



a1	a2	a3	a4	a5	a6	a7	a8	pa
b2	b3	b4	b5	b6	b7	b8	pb	b1
c3	c4	c5	c6	c7	c8	pc	c1	c2
d4	d5	d6	d7	d8	pd	d1	d2	d3

.....

disposizione Left Symmetric

analisi della affidabilità

- **Ipotesi:**
  - **i guasti nei vari dischi sono indipendenti fra loro**
  - **la possibilità di guasto è invariante nel tempo → distr. Exp. Intervallo fra 2 guasti**
- **le grandezze di interesse sono :**
  - **MTTF: mean time to failure**
  - **MTTR: mean time to repair**
  - **MTTDL: mean time to data loss**

affidabilità di un array di componenti uguali

- **Per un componente:**
  Pr[guasto al tempo x=t] = λ exp(- λt)
  E[x] = 1/ λ (attesa media guasto)

  Tasso di guasti = λ (cost.)
  $MTTF_C$ = 1/ λ

- **Per un array di N componenti:**
  Tasso di guasti = N $\times$ λ
  $MTTF_A$ = 1/ (Nλ) = $MTTF_C$ / N

analisi della affidabilità

- **RAID 0 (poco affidabili):**

  $MTTF_{RAID0} = MTTF_{DISCO} / N = MTTDL$

- **Es.:**

  $MTTF_{DISCO}$ = 30000 h > 3 anni

  con 100 dischi:

  $MTTF_{RAID0}$ = 30000/ 100 = 300 h ≈ 2 sett.

  con 8 dischi : 3759 h ≈ 22 sett.

analisi della affidabilità

- **RAID 1 - array di 2N dischi**

$$MTTF_{RAID1} = MTTF(\text{primo guasto}) / Pr[\text{secondo guasto entro MTTR}]$$

$$Pr[\text{secondo guasto entro MTTR}] = 1-\exp(-MTTR/MTTF_{DISCO}) \approx MTTR/MTTF_{DISCO}$$

$$MTTF_{RAID1} = MTTF_{DISCO}/(2N) \times MTTF_{DISCO}/MTTR$$

analisi della affidabilità

- **RAID 1 (molto affidabili):**

  $MTTF_{RAID1} = (MTTF_{DISCO})^2 / (2 \times N \times MTTR)$

- **MTTF(primo guasto) è basso**

  (es. con 16 dischi : ≈ 11 sett.)

ma

$MTTDL = MTTF_{RAID1}$ è elevatissimo:
si dovrebbero guastare contemporaneamente un disco e la sua copia per perdere dati!

analisi della affidabilità

- **RAID 5 - array di N dischi**

$$MTTF_{RAID5} = MTTF(\text{primo guasto}) / Pr[\text{secondo guasto entro MTTR}]$$

$$Pr[\text{secondo guasto entro MTTR}] = [1-\exp(-MTTR/MTTF_{DISCO})]\times(N-1) \approx (MTTR/MTTF_{DISCO})\times(N-1)$$

$$MTTF_{RAID1} = MTTF_{DISCO}/N \times MTTF_{DISCO}/(MTTR\times(N-1))$$

analisi della affidabilità

- **RAID 5 (molto affidabili):**

  $MTTDL_{RAID5} = (MTTF_{DISCO})^2 / (N \times (N-1) \times MTTR)$

- **Es.:**

  con 9 dischi e MTTR = 24 h : ≈ 60 anni

  molto inferiore al RAID 6 ma molto superiore ai dischi SLED (single large expensive disk)

Impiego dei RAID nei server

- **Esistono anche altre organizzazioni RAID oltre alla 5**
  (es. RAID6, RAID7, RAID10, RAID53, RAID0+1, RAID1+0)
- **L'organizzazione RAID5**
  (detta anche "il mirroring dei poveri")
  è molto usata, assieme al RAID6,
  in quanto sono ottimi compromessi fra prestazioni, affidabilità e complessità realizzativa

correzione degli errori (RAID6)

CODICI A RILEVAZIONE E CORREZIONE DI ERRORE

Data una parola binaria $C = (c_1, c_2, ..., c_k)$ (dove ogni c_i è un bit) è possibile effettuare un controllo di presenza di errore aggiungendo un bit c_{k+1} in modo tale che:

$$c_1 \oplus c_2 .... \oplus c_n \oplus c_{k+1} = 0.$$

dove $\oplus$ indica la somma modulo 2 .
c_{k+1} si chiama *bit di parità* poiché se i bit a 1 in C sono in numero pari c_{k+1} è = 0 , altrimenti e' =1.

Se la parità non torna vuol dire che almeno un bit è errato.

correzione degli errori

Questo sistema intercetta, quindi, tutti le C con un bit errato senza però dire quale è.
Gli errori su più bit possono compensarsi.

Esistono sistemi migliori che però usano più bit di ridondanza per il controllo degli errori.

Un *codice binario lineare* C(n,k) è un sottospazio di dimensione k dello spazio vettoriale $\{0,1\}^n$ di tutte le parole binarie di lunghezza n.
I 2^k elementi di C(n,k) sono chiamati parole di codice (*codewords*) ed in generale possono essere scritte come:

$$C = (c_1, c_2, \ldots, c_n),$$ **dove ogni** c_i **è un bit.**

correzione degli errori

Un codice binario lineare C(n,k) puo' essere descritto da un sistema di m= n-k equazioni lineari indipendenti. La matrice H, (n-k)×n, di questo sistema e' chiamata *check matrix* del codice.

Se H e' la check matrix di C(n, k), allora ogni codeword C ha la propieta' che :

$$HC^T = |0|^T$$

(dove l'apice T indica la trasposizione e le operazioni interne al prodotto di matrice sono booleane: somma ⊕ e prodotto logico ⊗).

correzione degli errori

In generale, per ogni parola A in $\{0,1\}^n$ la parola :

$y = (y_1, y_2, \ldots, y_{m=n-k})$, data da: $y^T = HA^T$,

è chiamata *sindrome* di A.

A causa della dimensione m di y , il numero di possibili sindromi è 2^m.
Se la sindrome corrisponde al numero 0 in codifica binaria allora la parola è *corretta* (appartiene al codice), altrimenti il numero binario espresso da y dà la *posizione del bit errato*.
Se i bit errati sono più di uno si rileva soltanto che C è in errore ma non si è in grado di correggerla.

correzione degli errori

Consideriamo una matrice di m righe , allora $n= 2^m-1$ sono le colonne per avere un sistema di equazioni lineari indipendenti costruite in modo da non avere mai due colonne la cui somma termine a termine dia sempre 0.

Ad esempio: per m=3 (k=4), ed $n= 2^3-1=7$ si può avere una matrice H semplicemente enumerando tutti i possibili numeri di m bit (escluso lo 0):

$$H = \begin{matrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix}$$

correzione degli errori

supponiamo di voler memorizzare il dato (di k=4 bit) 1 1 0 0 e di prendere come *bit di controllo* per la parola C da codificare i bit c_1, c_2 e c_4 allora :

$C = c_1, c_2, 1, c_4, 1, 0, 0$

calcoliamo adesso c_1 verificando la regola di parità con l'ultima riga H(3) di H :

$H(3)C^T = c_1\ 0\ 1\ 0\ 1\ 0\ 0 = 0$

quindi $c_1=0$, analogamente,

$H(2)C^T = 0\ c_2\ 1\ 0\ 0\ 0\ 0 = 0$ quindi $c_2=1$ e

$H(1)C^T = 0\ 0\ 0\ c_3\ 1\ 0\ 0 = 0$ quindi $c_3=1$

C diventa quindi : 0 1 1 1 1 0 0

correzione degli errori

Nel caso in cui un bit diventi errato , ad esempio il 5º,
C' diventa 0 1 1 1 0 0 0,
la moltiplicazione HC'^T dà una sindrome
(1 0 1) che corrisponde in H alla 5ª colonna e
quindi è errato il quinto bit.

Quindi H ci avverte che il 5º bit è errato e può essere corretto invertendolo!

Scambiando le colonne il sistema non cambia di significato, le H ed i bit di controllo sono diversi e la sindrome è diversa ed indica che è sbagliato il bit corrispondente alla colonna di H uguale alla sindrome

correzione degli errori

È importante notare che i tre bit ridondanti (di parità) c_1, c_2 e c_4 sono stati scelti in corrispondenza delle colonne di H di peso 1,
H può essere riscritta per maggior chiarezza come:

$$H = \left(\begin{array}{ccc|cccc} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right) \quad o \quad \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right)$$

dove la parte 3×3 iniziale della matrice e' a diagonale unitaria .
Tre bit di parità servono quindi per il controllo di informazioni di 4 bit.

correzione degli errori

Il tipo di codice si chiama codice *Hamming* (7,4).
In generale il codice *Hamming* è (n,k), con
$n=2^m-1$ e $k=2^m-1-m$.

Il codice Hamming è uno dei più semplici,
ne esistono di molto più complessi che offrono
prestazioni superiori per la rilevazione e la
correzione di errori

(il codice Reed-Solomon usato nel RAID6
è uno di questi).