

Notare come...

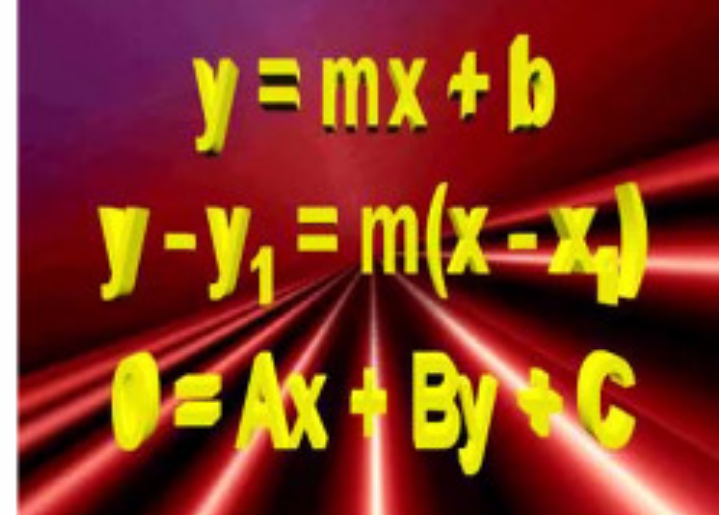
- ◆ ... gli Hamming sono una generalizzazione dei codici che abbiamo visto prima (parity bit, repetition)...
- ◆ ... e possiamo vederlo meglio quando li consideriamo appunto nella grande famiglia dei ***codici lineari***.

$$y = mx + b$$

$$y - y_1 = m(x - x_1)$$

$$0 = Ax + By + C$$

Esempio: i parity bit...



- ◆ Il parity bit 3 (ogni tre bit, inseriamo un parity bit P):
- ◆ **$(a \ b \ c) \rightarrow (a \ b \ c \ P)$**
- ◆ Possiamo usare la matrice generatrice
 $\begin{matrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{matrix}$
- ◆ Notare, peso minimo 2 \rightarrow codice **$(4,3,2)$**

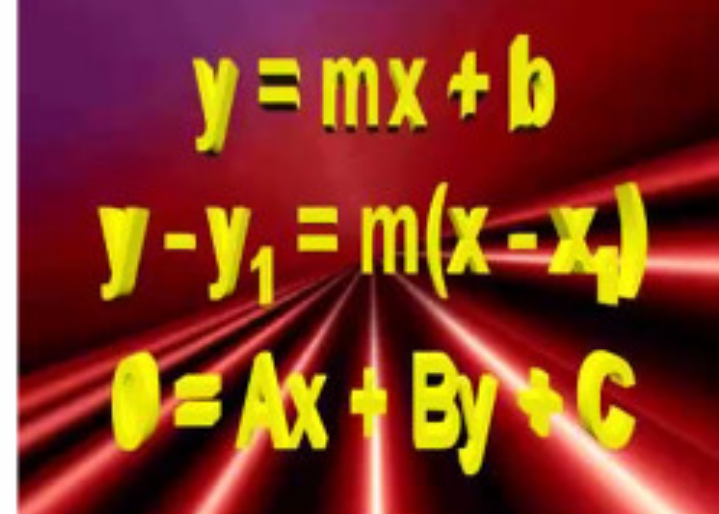
Esempio: i repetition codes...

◆ R3 ("balbettiamo" tre volte)

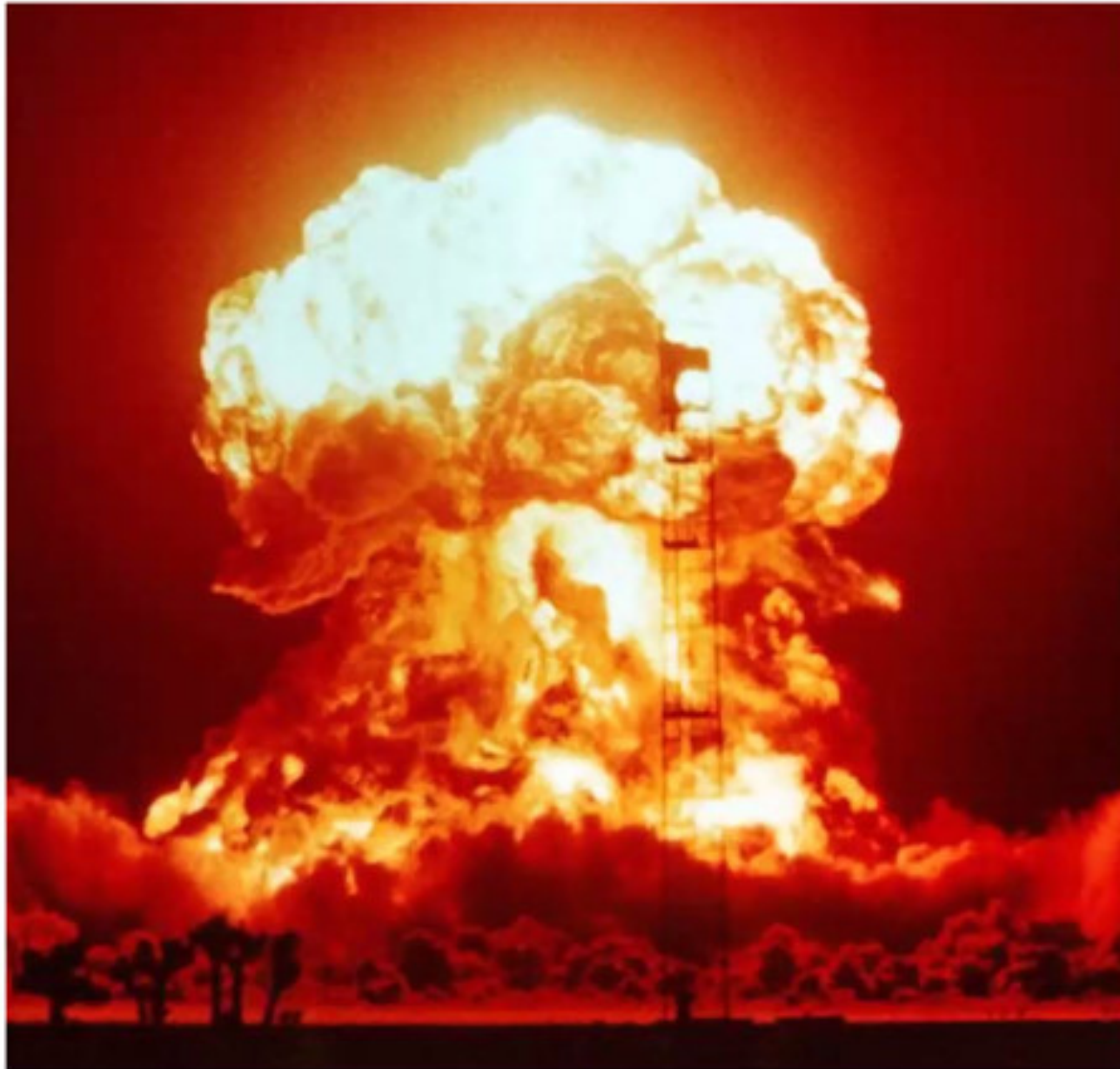
◆ **(a) → (a a a)**

◆ Possiamo usare la matrice generatrice
1 1 1

◆ Notare, peso minimo 3 → codice **(3,1,3)**

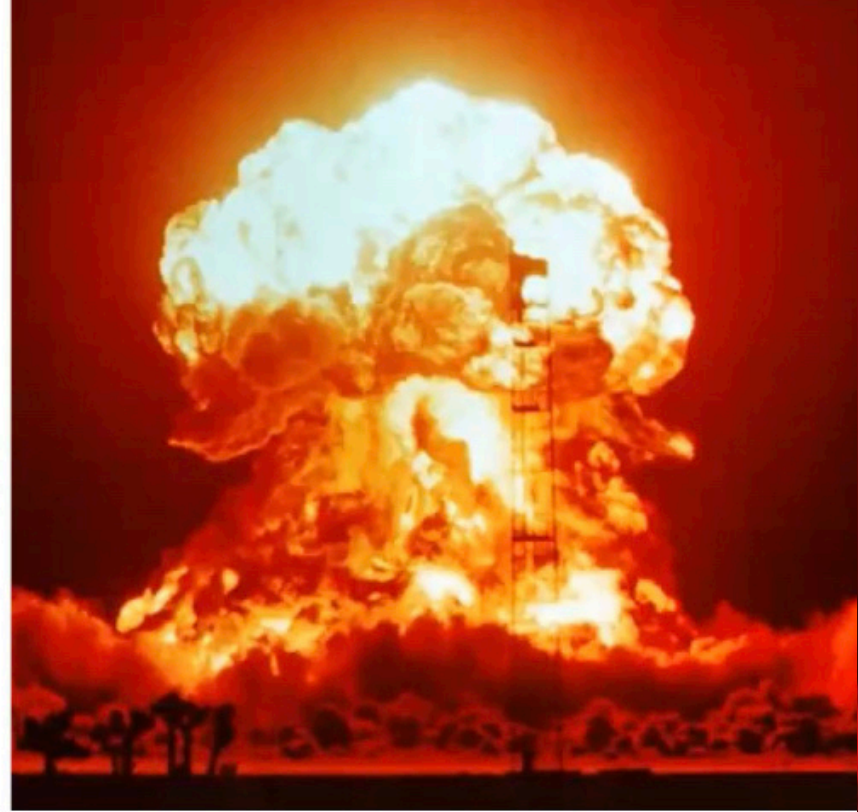


Vediamo ora i problemi pratici
con gli errori...



I burst

- ◆ Nella pratica, in molti casi gli errori non sono del tutto casuali, ma occorrono in ***burst*** (“esplosioni” ravvicinate)
- ◆ Bene o male?



I burst

- ◆ Nella pratica, in molti casi gli errori non sono del tutto casuali, ma occorrono in ***burst*** (“esplosioni” ravvicinate)



MALE!!!!

- ◆ Questo crea ***problemi mostruosi*** ad ogni codice error-correcting...!!!



E quindi...?

- ◆ Potremmo tenerne conto e anzi ***sfruttare la loro struttura*** a nostro vantaggio!
- ◆ Come????









Il metodo della matrice invertita (*interleaving*)



Char.

ASCII

Check bits

H	1001000
a	1100001
m	1101101
m	1101101
i	1101001
n	1101110
g	1100111
	0100000
c	1100011
o	1101111
d	1100100
e	1100101

00110010000
10111001001
11101010101
11101010101
01101011001
01101010110
01111001111
10011000000
11111000011
10101011111
11111001100
00111000101

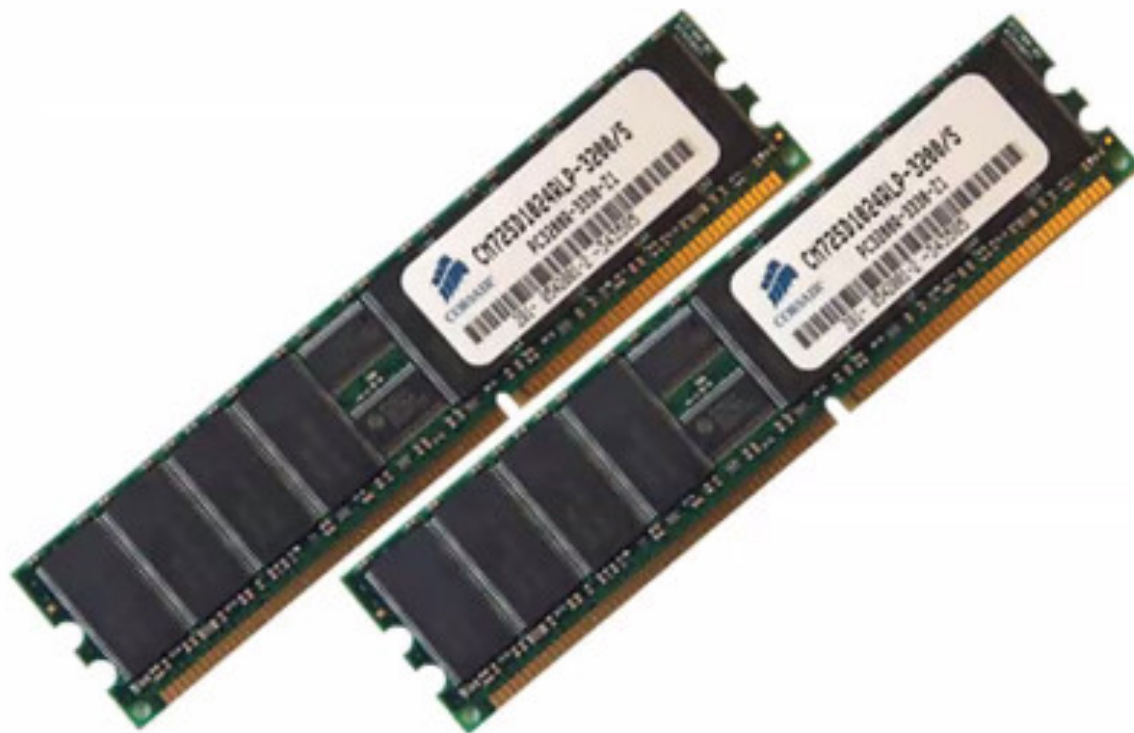
Order of bit transmission

Il nostro computer!

- ◆ Nel suo piccolo, c'è anche qui ***comunicazione dei dati***:
dal processore alla RAM al disco rigido
etc etc
- ◆ Però, le componenti sono molto più affidabili che non una rete esterna, quindi non dobbiamo preoccuparci

RAM

◆ Consideriamo ad esempio la RAM



Unità di misura

- ◆ Ci serve un'unità di misura grossetta: ad esempio, il **quadrilione**

◆ 1000000000000000

◆ $= 10^{15}$

Dati del nostro computer

- ◆ RAM che va a 1600MhZ
- ◆ 1Gb di RAM
- ◆ Supponiamo, la probabilità di un errore nella RAM sia bassissima: uno ogni ***cento quadrilioni (!)***
- ◆ Ogni quanto in media ci sarà un errore?



Dati in RAM



◆ → un errore ogni

0.08 secondi
(!!!!!!!)

E un doppio errore?



- ◆ Senza tenere conto dei burst, un doppio errore circa ogni **8 quadrilioni di secondi (!!)**
- ◆ → **scelta di design:** se vogliamo migliorare il nostro sistema, dovremmo tenere ***ben conto*** degli errori di ***potenza 1***, e possiamo tralasciare quelli di ***potenza 2***

E per i dischi rigidi?

- ◆ La situazione non cambia di molto (fate il conto): l'access rate è di certo molto minore della RAM, ma non di molti ordini di grandezza, il che significa che ci saranno errori in termini di anni invece che di secondi



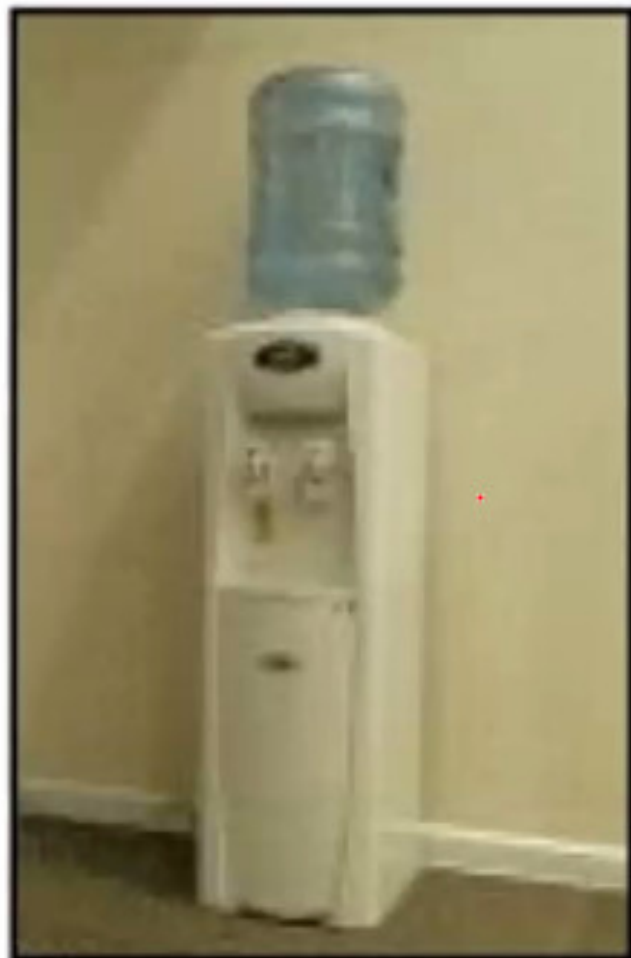
E allora?



RAID

- ◆ **R**edundant **A**rray of **I**nexpensive (o
talvolta "Independent") **D**isks

RAID



Standalone



Cluster

RAID



Hot swap



RAID 0

RAID



RAID 1



RAID 5

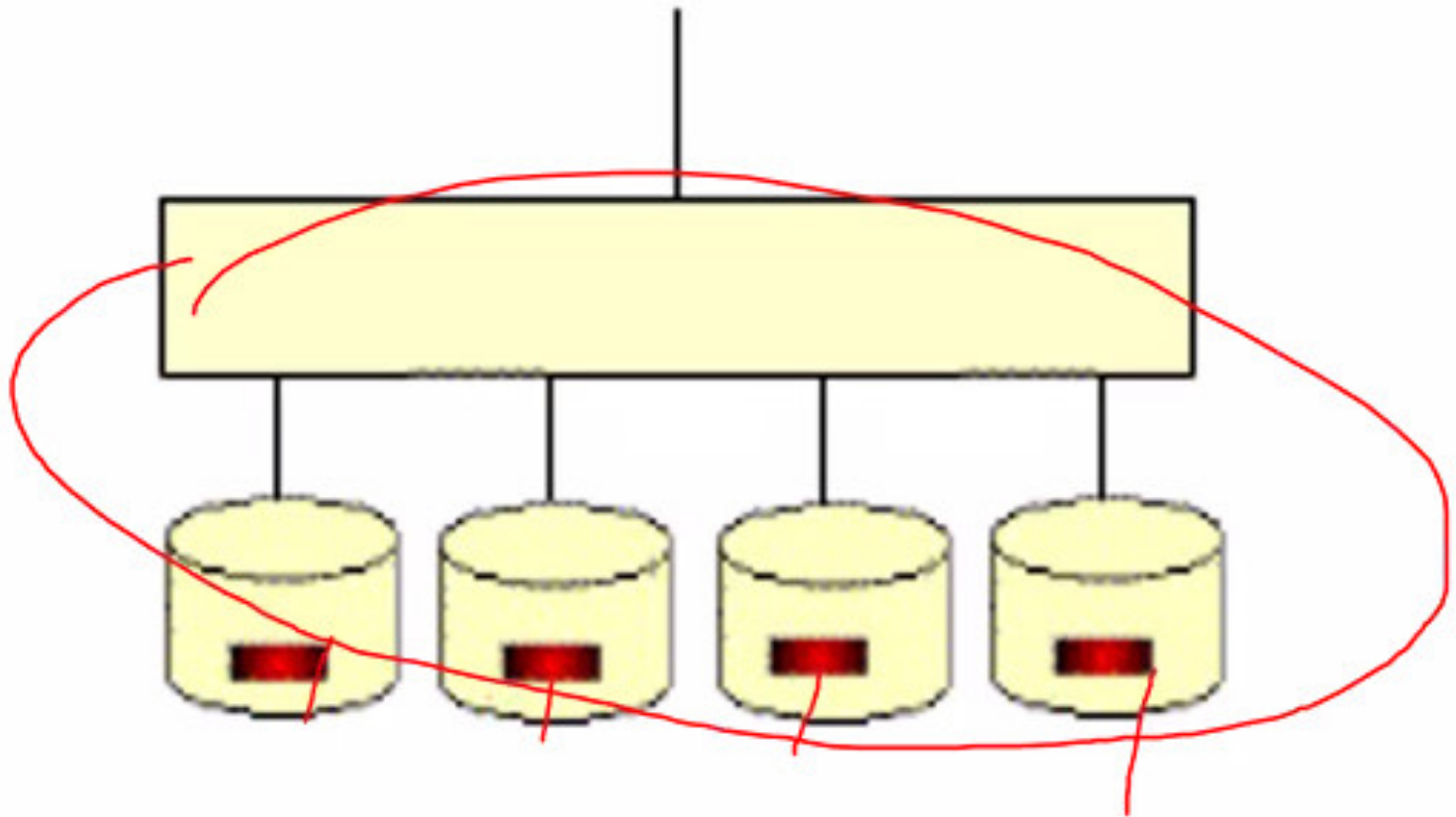
RAID



RAID 0+1

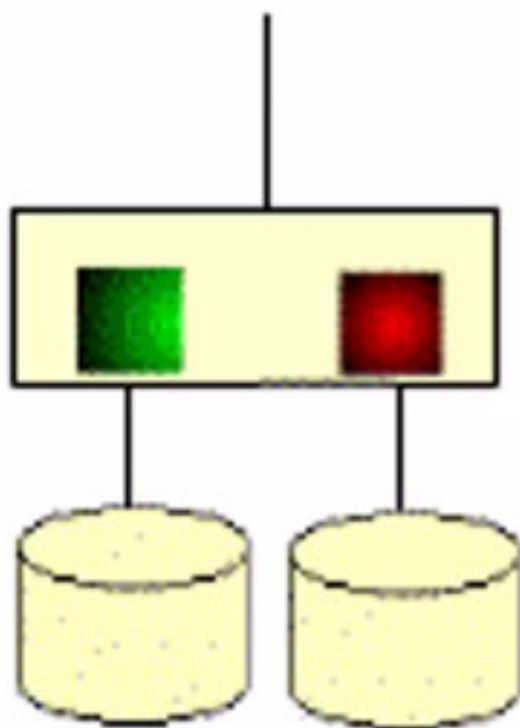
RAID 0

◆ Overlay (striping), senza codici di errore



RAID 1

◆ Il codice R2 (!)

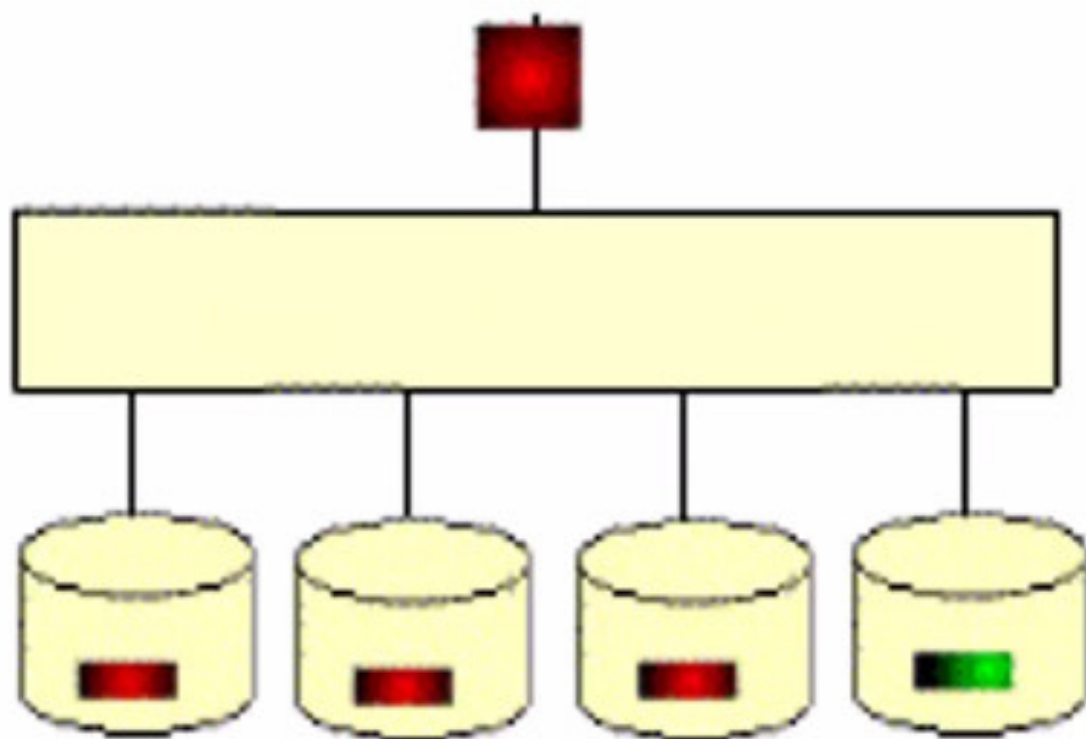


RAID 2

- ◆ Overlay a livello di bit con un ***Hamming code*** (!)
- ◆ Dà un ***altissimo*** livello di protezione, ma costa molto (per ora)

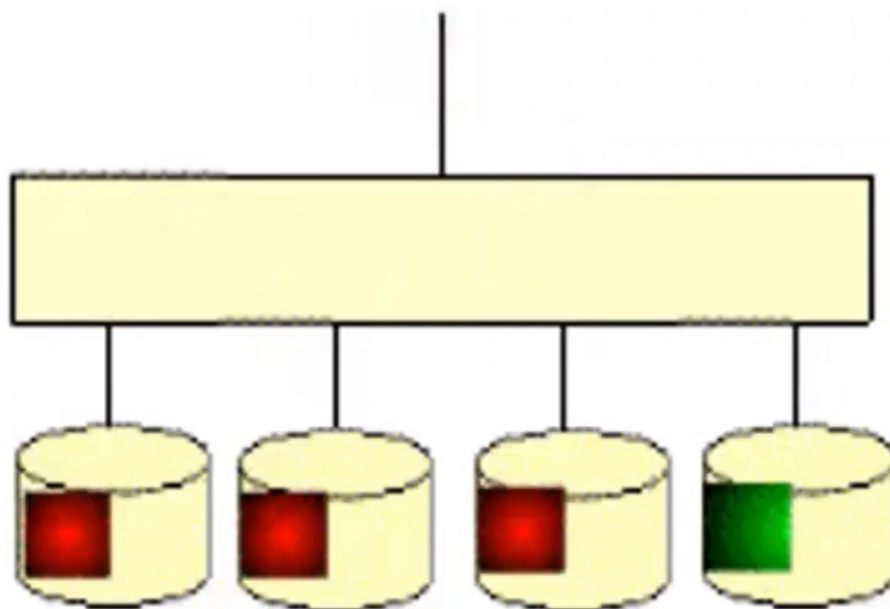
RAID 3

- ◆ Overlay con il codice parity



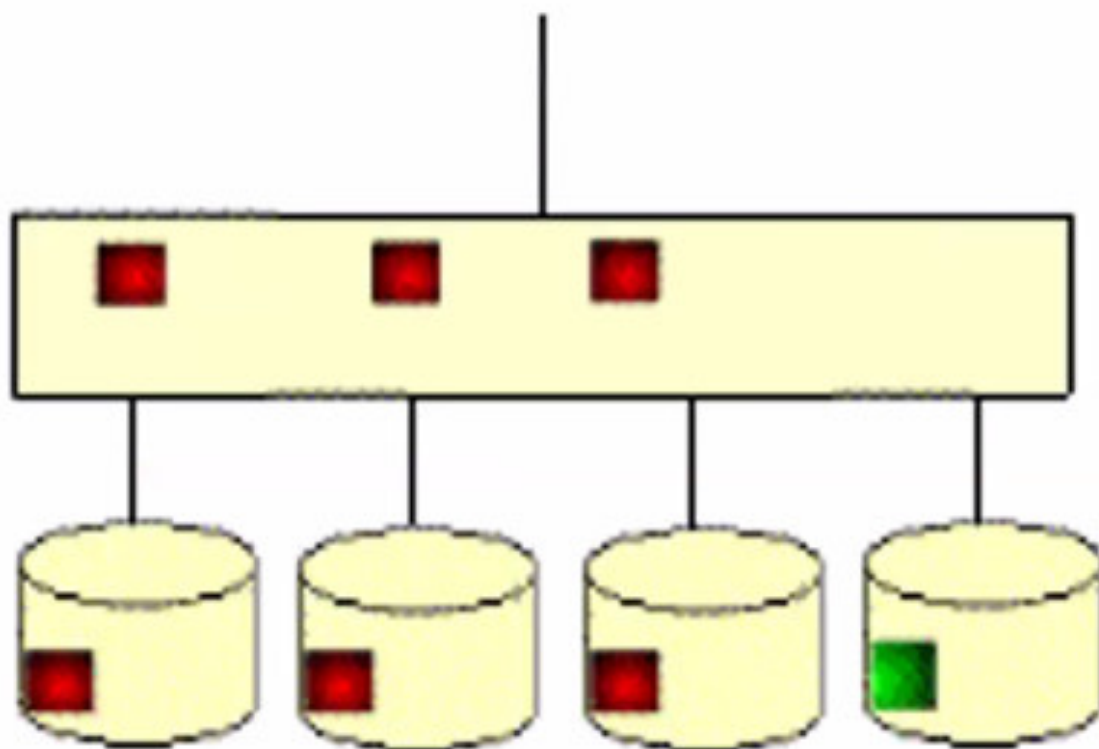
RAID 4

◆ Codice parity



RAID 5

◆ Codice parity distribuito



E cosi' via...

◆ RAID 6, RAID 7, RAID 1+0, RAID 50,
RAID 51, RAID 53, RAID 100, RAID
3+0, RAID 0+1, RAID-S, RAID 3+1,
RAID 7+1 ...

E la RAM?



ECC RAM



- ◆ Usa un codice di ***error correction***, variante di quello di Hamming, di tipo **(72, 64)**
- ◆ Se vi siete mai chiesti ***perché*** le RAM di nuova generazione non andavano più veloci delle vecchie: hanno una ***penalità*** dell'**11%** (il data rate è l'89%)

SECCDED



- ◆ Attenzione che la scheda madre deve supportare la funzionalità:
- ◆ Anche se trovate scritto nelle specifiche che “supporta le ECC”, significa solo che potete usarle, ma NON NECESSARIAMENTE che ci sia error correction
- ◆ → deve supportare **SECCDED**:
Single Error Correction,
Double Error Detection

Esempio paradossale... le
schede di fascia alta... (!)



Altro esempio...

- ◆ Le schede di memoria...
- ◆ ECC con Hamming!



Andiamo oltre l'atmosfera...

