

Dischi RAID

storage
high-performance
high-reliability

11/03/06

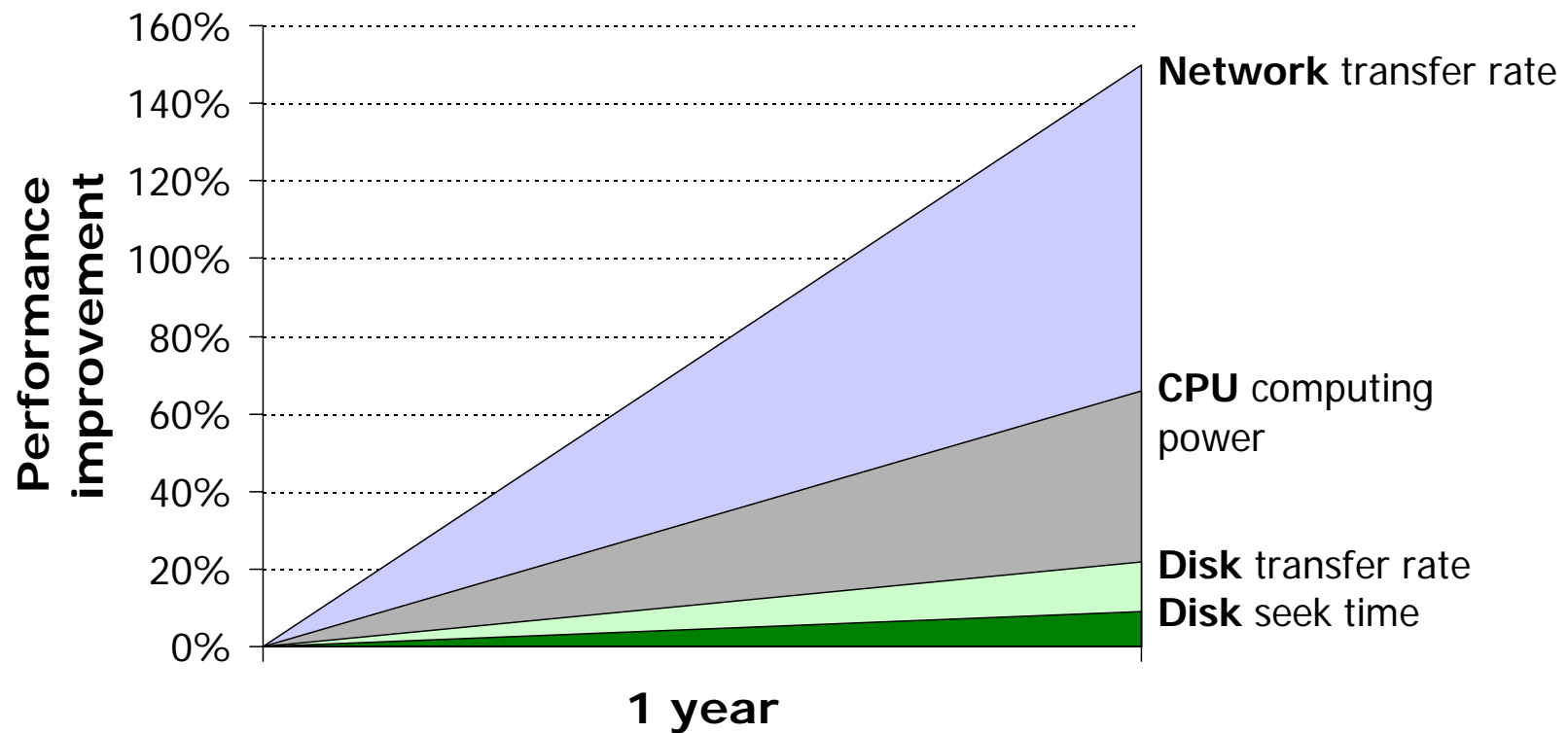
indice

- caratteristiche generali dei dischi
- prestazioni
- tecniche per migliorare le prestazioni di I/O
- dischi RAID
 - tipologie di RAID
 - affidabilità

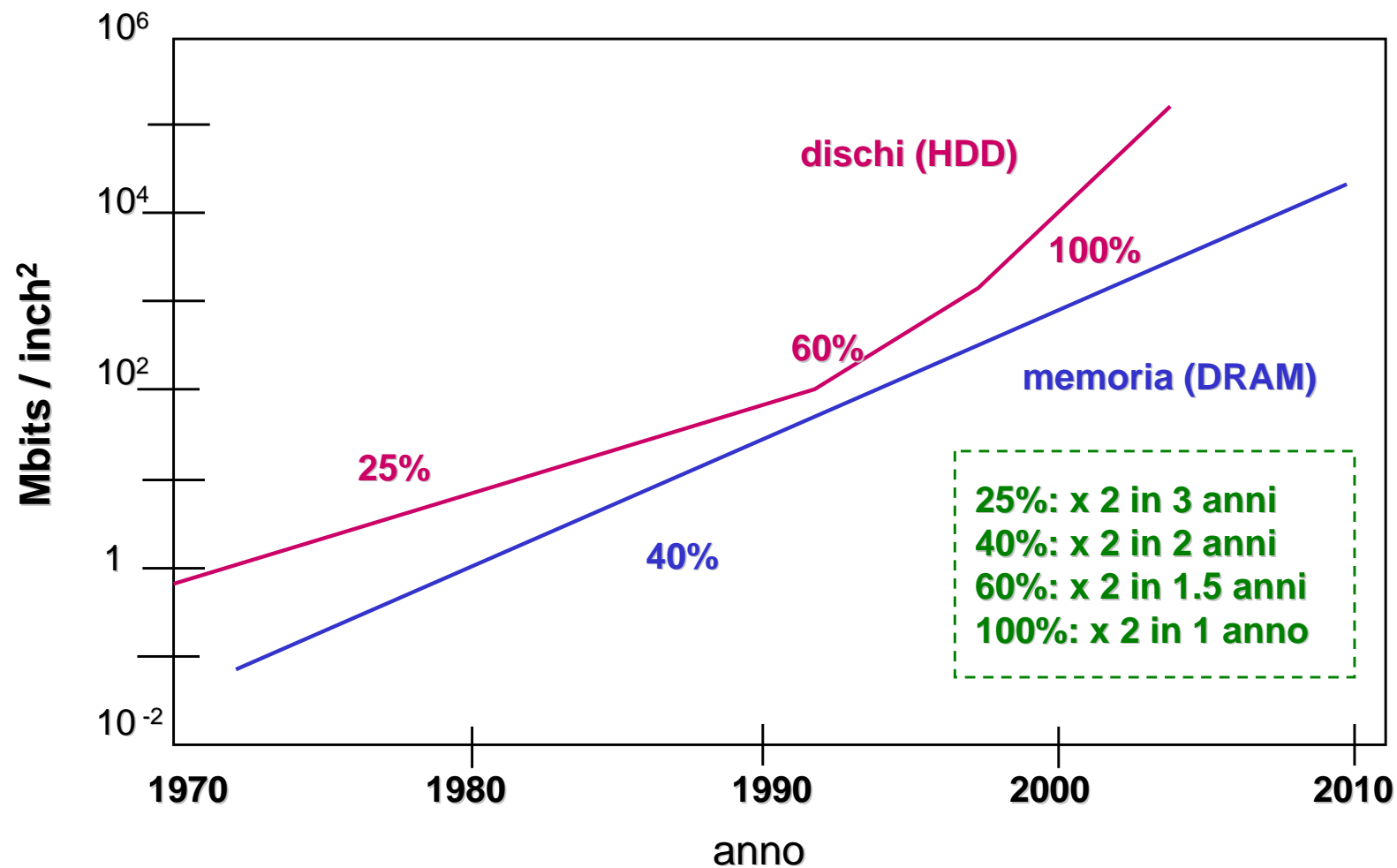
dischi: caratteristiche generali

evoluzione delle capacità (legge di Moore)

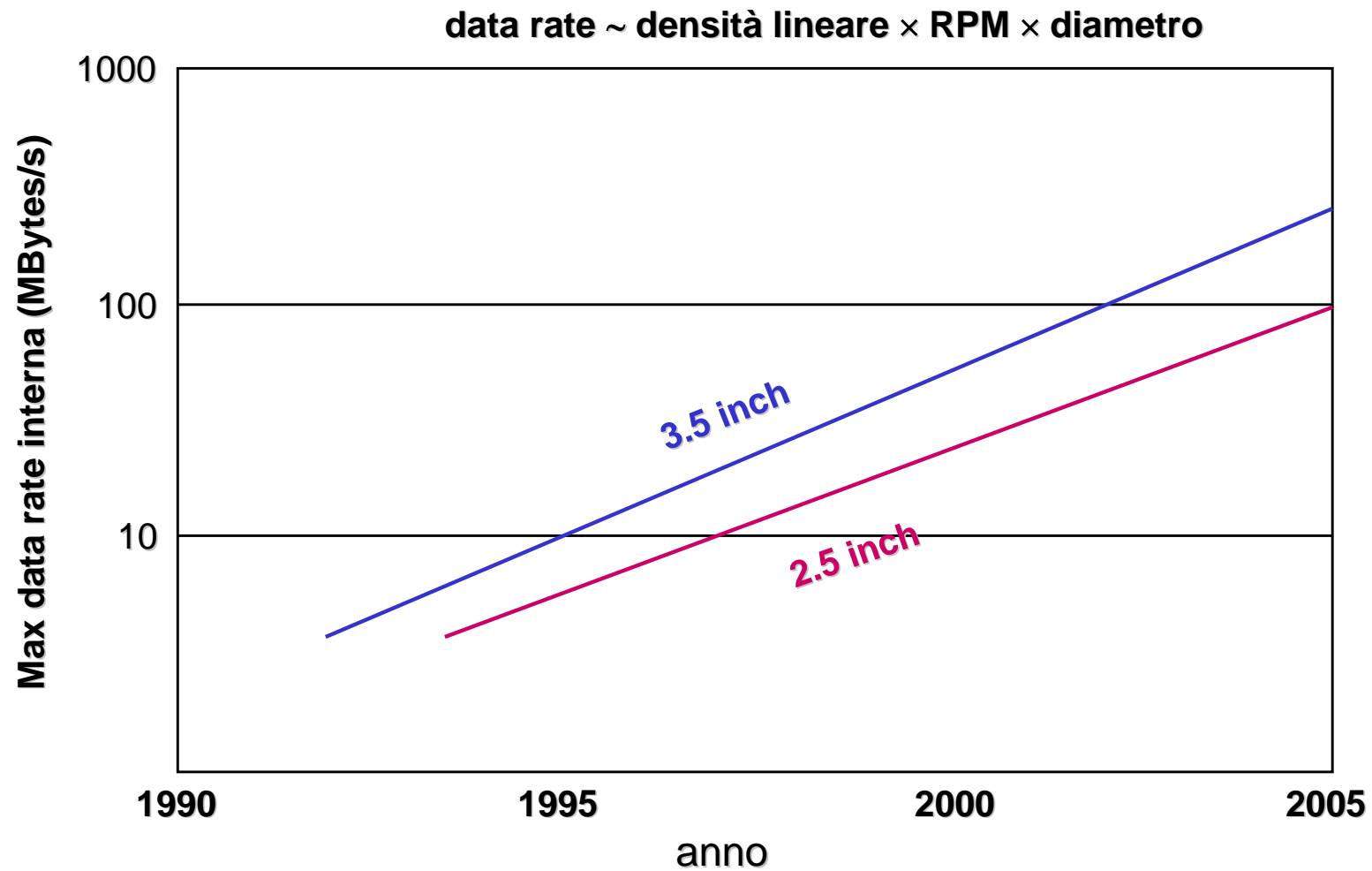
- legge empirica che all'origine riguardava l'andamento della densità dei transistori per chip (che raddoppia ogni 18 mesi)
- concerne l'incremento della *capacità operativa* (che si moltiplica per 100 ogni 10 anni)



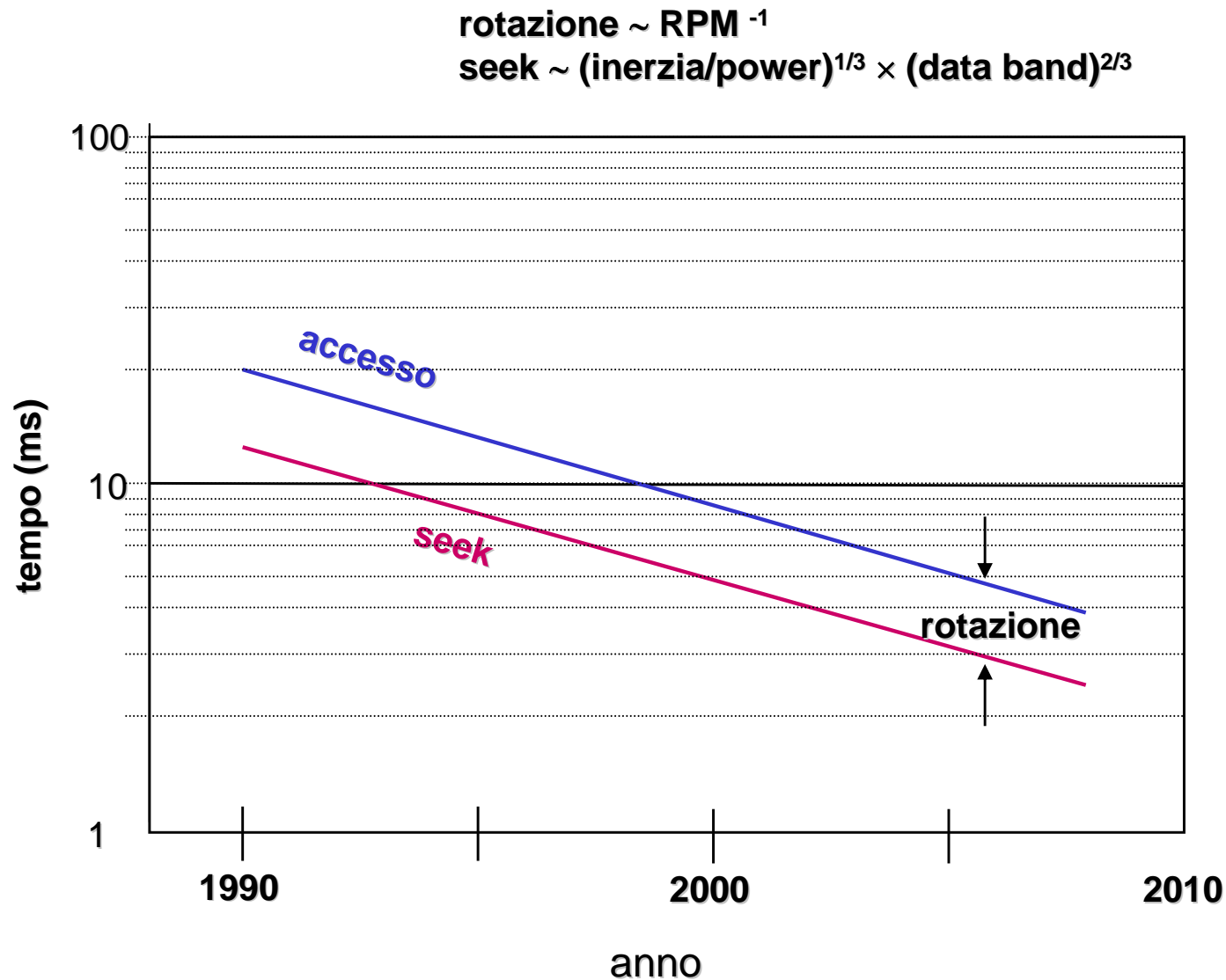
esempio: evoluzione dischi (densità superficiale)



esempio: evoluzione dischi (data rate)

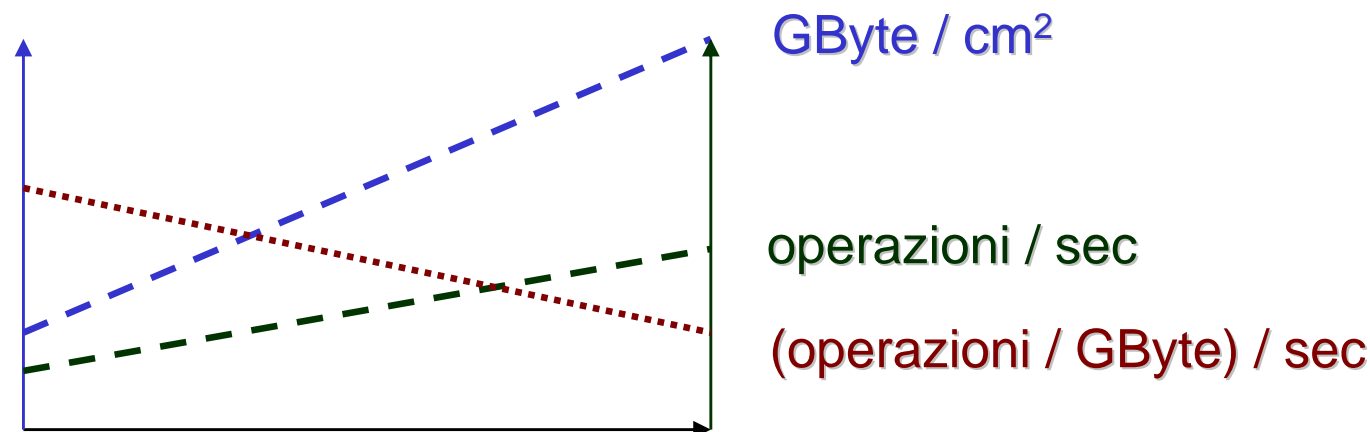


esempio: evoluzione dischi (tempi di accesso)

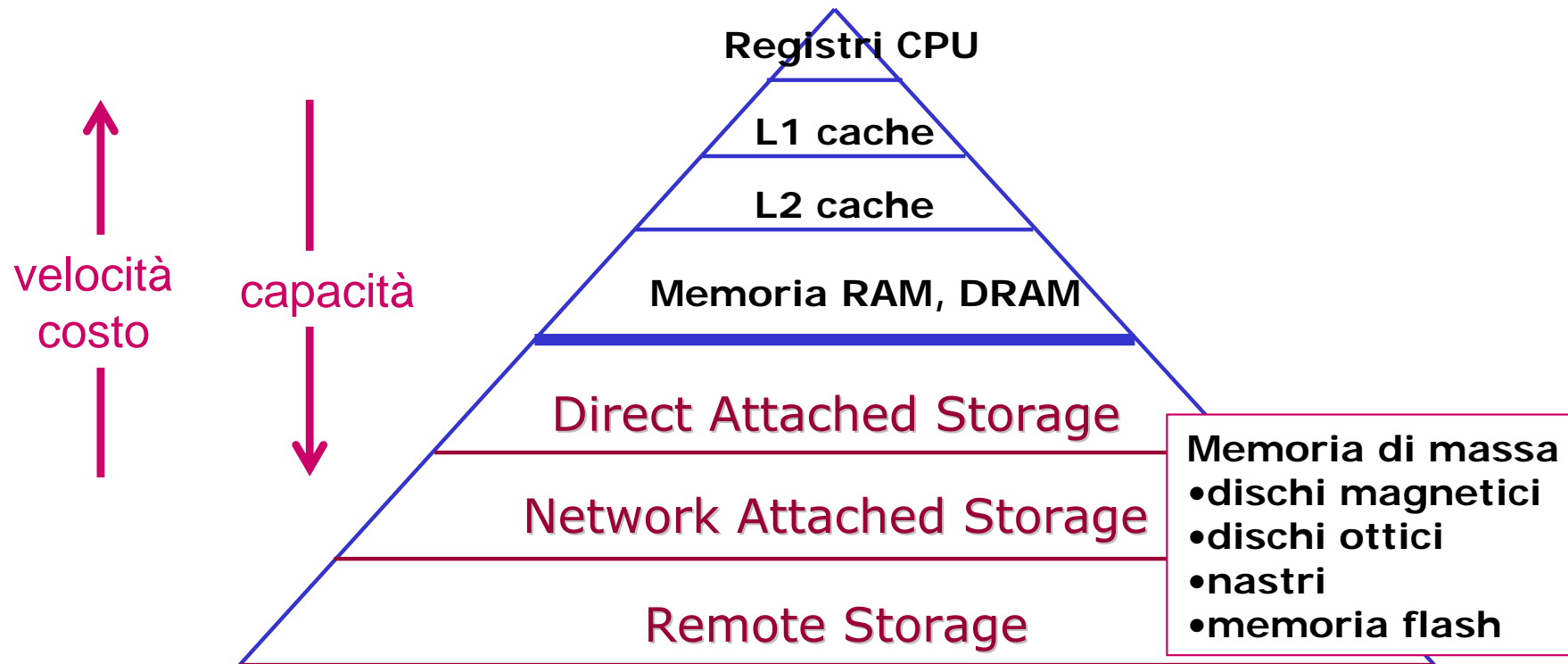


osservazione

- la crescita con **tassi diversi** di alcune grandezze può dare luogo ad alcuni problemi:
 - in particolare la capacità di memoria cresce più rapidamente dei tempi di accesso perciò la densità degli accessi è diminuita nel tempo
 - pericolo di non completo utilizzo (inedia o "*starvation*") dei dispositivi (processori e dischi)
- la capacità dei dischi è cresciuta dal 1956 di 5×10^7 volte,
- entro 4 anni si pensa di raggiungere 500 Gbit per inch²
- con metodi olografici 1 Tbit può essere contenuto in un volume di 1 cm³
(fonte: *Scientific American* aug. 05)



livelli gerarchici di memoria



livelli gerarchici di memoria (2)

Esempio di calcolo del tempo medio di accesso al dato

	<i>layer</i>	<i>tempo t</i>	<i>miss rate m</i>	<i>prob. p</i>	<i>p x t</i>
1	<i>Reg</i>	1,00E+00	1,00E-01	1,00E+00	1,00E+00
2	<i>L1</i>	1,00E+00	5,00E-02	1,00E-01	1,00E-01
3	<i>L2</i>	8,00E+00	2,00E-02	5,00E-03	4,00E-02
4	<i>Main mem.</i>	1,00E+02	1,00E-01	1,00E-04	1,00E-02
5	<i>Local disk</i>	1,00E+07	2,00E-02	1,00E-05	1,00E+02
6	<i>Net server</i>	5,00E+07	2,00E-02	2,00E-07	1,00E+01
7	<i>Remote server</i>	4,00E+08	0,00E+00	4,00E-09	1,60E+00
<i>tot</i>					112,75

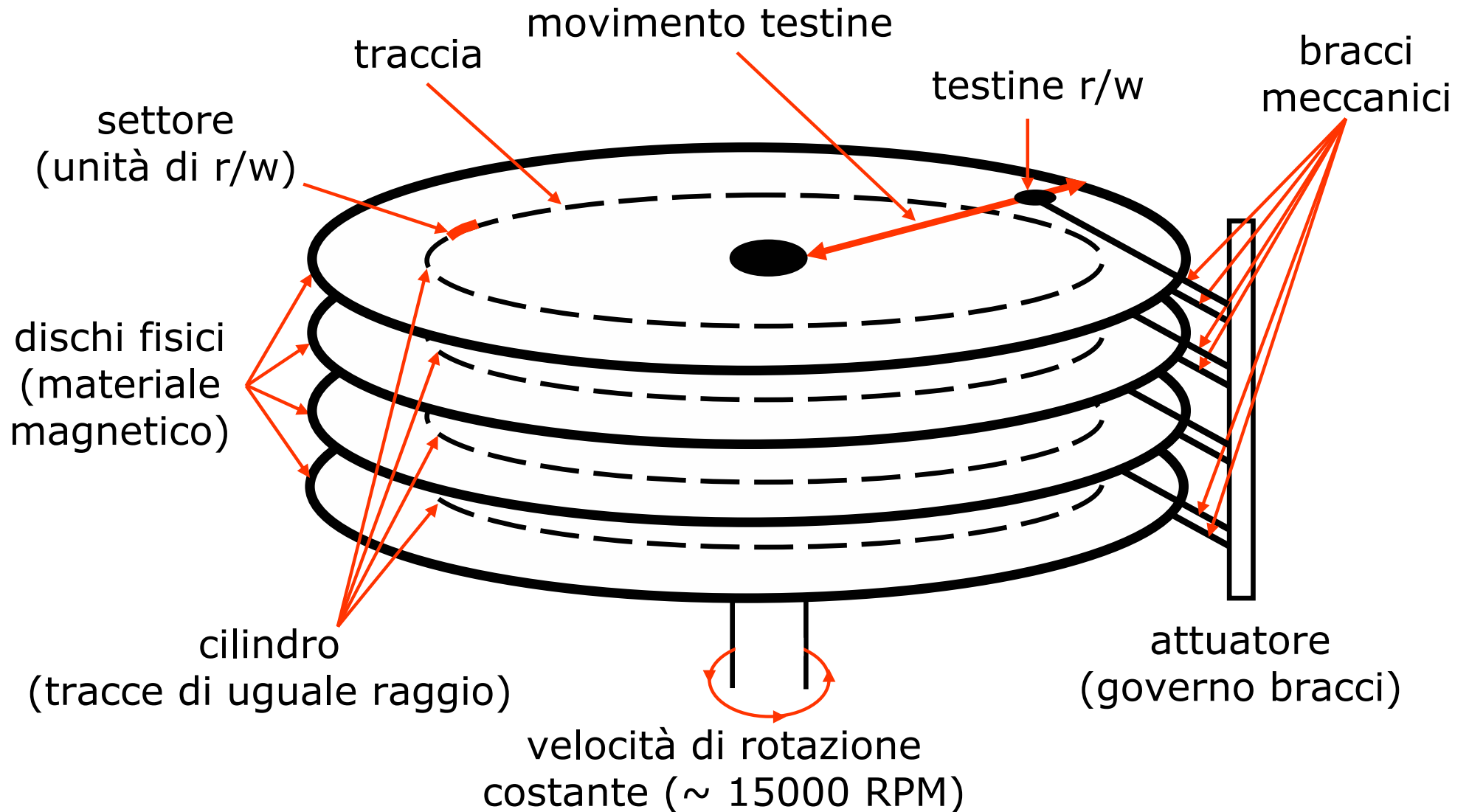
$$\begin{aligned}
 p(i) &= p(i-1) \times m(i-1) && \text{probabilità di accesso al livello (i)} \\
 p(i) \times t(i) &&& \text{tempo di accesso al livello (i)} \\
 \text{tempo totale} &= \sum p(i) \times t(i)
 \end{aligned}$$

esercizio

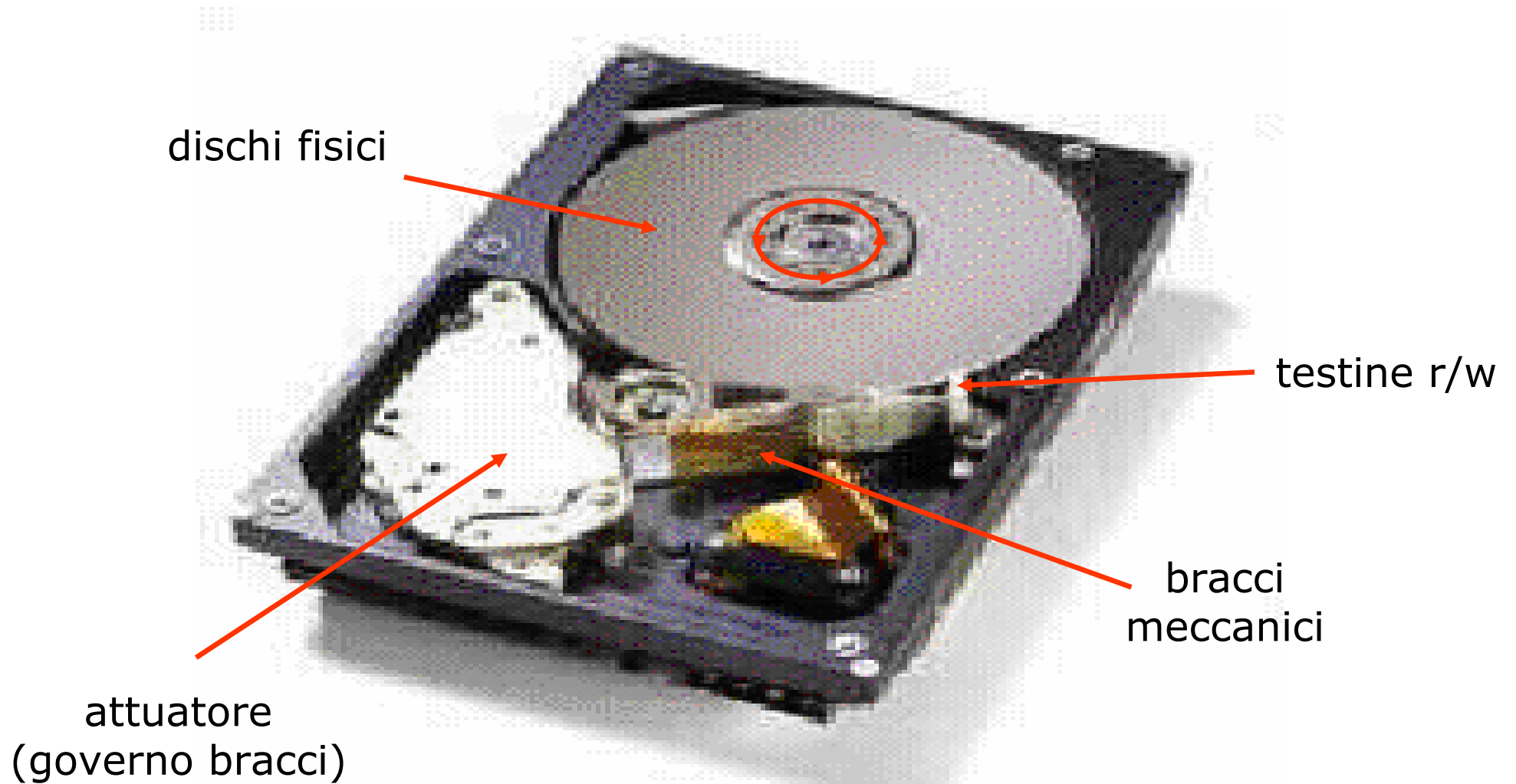
	<i>layer</i>	<i>tempo t</i>	<i>miss rate m</i>	<i>prob. p</i>	<i>p x t</i>
1	<i>Reg</i>	1,00E+00	1,00E-01	1,00E+00	1,00E+00
2	<i>L1</i>	1,00E+00	5,50E-02	1,00E-01	1,00E-01
3	<i>L2</i>	8,00E+00	2,20E-02	5,50E-03	4,40E-02
4	<i>Main mem.</i>	1,00E+02	1,10E-01	1,21E-04	1,21E-02
5	<i>Local disk</i>	1,00E+07	2,20E-02	1,33E-05	1,33E+02
6	<i>Net server</i>	5,00E+07	2,20E-02	2,93E-07	1,46E+01
7	<i>Remote server</i>	4,00E+08	0,00E+00	6,44E-09	2,58E+00
<i>tot</i>					151,47

Ecco come si modificano i tempi al variare delle miss rate

dischi magnetici



unità disco



un disco: pila di piatti

- diametro di circa 9 cm (3,5 in)- ciascuno con due facce
- ruotanti da 7200 a 15000 giri/minuto
- ogni faccia contiene 10000-50000 tracce concentriche
- divise in settori (100-500) tipicamente di 512 byte
- che contengono numero settore - gap- informazioni con sequenza di correzione errori - gap - numero prossimo settore e così via
- inizialmente ogni traccia aveva lo stesso numero di settori (quindi di bit) ma con ZRB (zone bit recording) il numero dei settori varia e resta invece costante lo spazio fra bit successivi, perciò spostandosi verso l'esterno le tracce aumentano la capacità

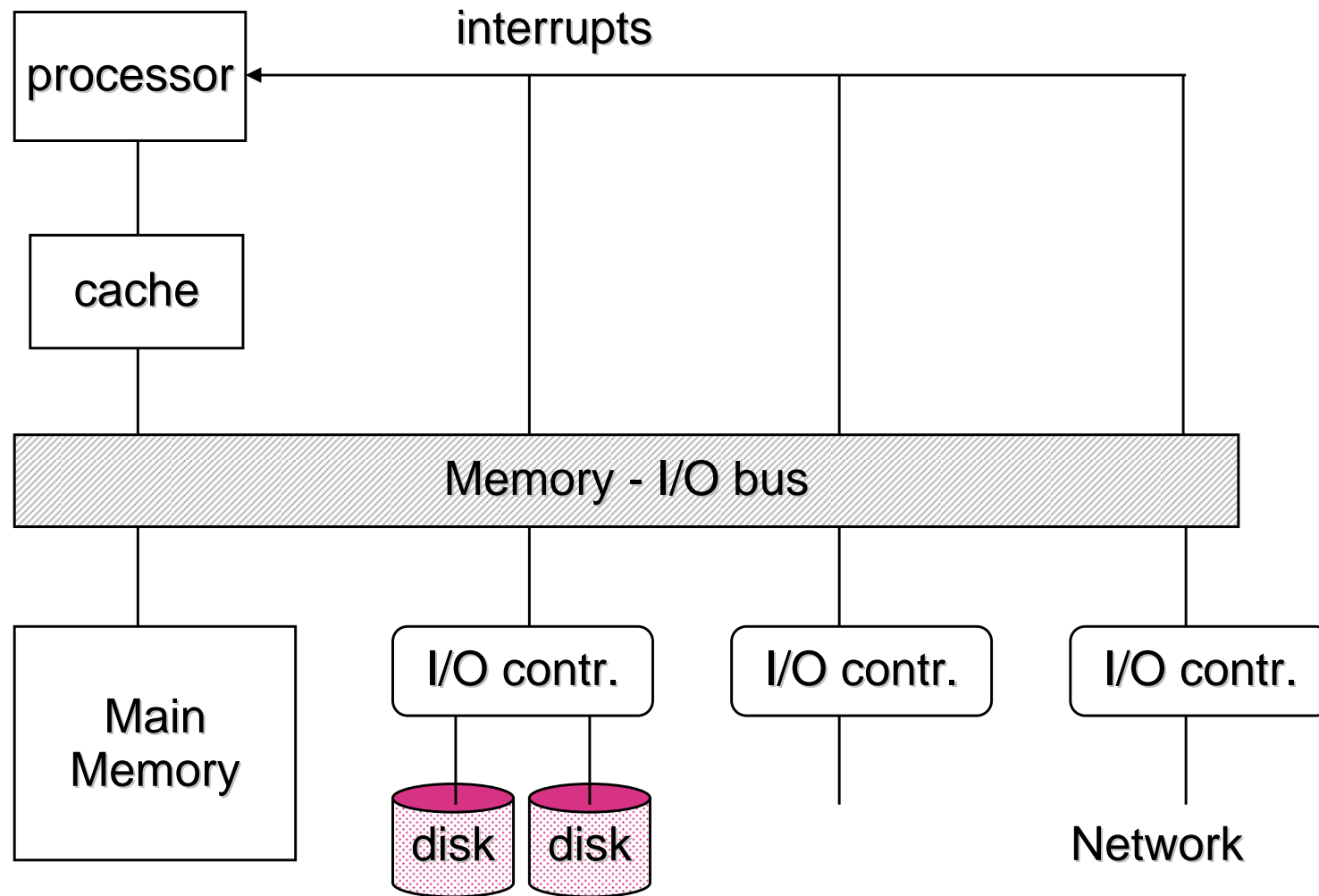
caratteristiche di un disco magnetico

- le *testine* di lettura/scrittura
 - si muovono insieme e si trovano contemporaneamente sulla stessa traccia per ogni superficie
 - *cilindro*: insieme delle tracce di tutte le facce sotto le testine posizionate in un determinato punto
 - *seek*: operazione meccanica di posizionamento della testina, la sua durata dipende dalle tracce attraversate, mediamente 3-14 ms
- *trasferimento*: la sua durata è il tempo necessario per leggere/scrivere un blocco
 - dipende da: dimensione del settore, velocità di rotazione e densità dei dati sul supporto, 30-80 MB/sec.
 - buona parte delle unità di controllo dei dischi hanno una cache (velocità di trasferimento fino a 320 MB/sec)

caratteristiche indicative di dischi (2004)

Characteristics	Seagate ST373453	Seagate ST3200822	Seagate ST94811A
<i>Disk diameter (inches)</i>	2.50	3.50	3.50
<i>Formatted data capacity (GB)</i>	73.4	200	40.0
<i>Cylinders</i>	31310		
<i>Sectors per drive</i>	143,374,744	390,721,968 (LBA mode)	78,140,160 (LBA mode)
<i>Number of disk surfaces (heads)</i>	8	4	2
<i>Rotation speed (RPM)</i>	15,000	7200	5400
<i>Internal disk cache size (MB)</i>	8	8	8
<i>External interface, bandwidth (MB/sec)</i>	Ultra320 SCSI, 320	Serial ATA, 150	Ultra ATA, 100
<i>Sustained transfer rate (MB/sec)</i>	57-86	32-58	34
<i>Minimum seek (read/write) (ms)</i>	0.2/0.4	1.0/1.2	1.5/2.0
<i>Average seek (read/write) (ms)</i>	3.6/4.0	8.5/9.5	12.0/14.0
<i>Mean time to failure (MTTF) hours</i>	1,200,000@25 °C	600,000@25 °C	330,000@25 °C
<i>Warranty (years)</i>	5	3	-
<i>Nonrecoverable read error per bit read</i>	< 1 per 10 ¹⁵	< 1 per 10 ¹⁴	< 1 per 10 ¹⁴
<i>Price in 2004 (\$/GB)</i>	\$5	\$0.5	\$2.5

sistemi di I/O



bus

- *bus*: link di comunicazione condiviso che usa un insieme di “fili” per connettere sottosistemi multipli
- *vantaggi*:
 - basso costo
 - versatilità
- unico schema di connessione
 - si possono aggiungere nuovi dispositivi
 - le periferiche che usano lo stesso tipo di bus possono essere spostate fra sistemi diversi
- *svantaggi*:
 - collo di bottiglia delle comunicazioni, la larghezza di banda limita il massimo throughput I/O
 - lunghezza del bus
 - numero di dispositivi
- *prospettive*:
 - interconnessioni seriali ad alta velocità con switch

bus (2)

■ tipi di bus

- *processore - memoria* (corti e ad alta velocità)
- di *I/O* (lunghi con ampio range di banda)
- *backplane* (permette la coesistenza di memoria e I/O)
- sono organizzati in modo gerarchico

■ comunicazione

- *sincrona* (generalmente usata dal bus di memoria); i dispositivi devono avere lo stesso ritmo di clock
- *asincrona* (usata dai bus di I/O); il coordinamento della trasmissione è gestito da un protocollo di *handshaking*; richiede linee aggiuntive per i segnali di controllo

■ trasferimento

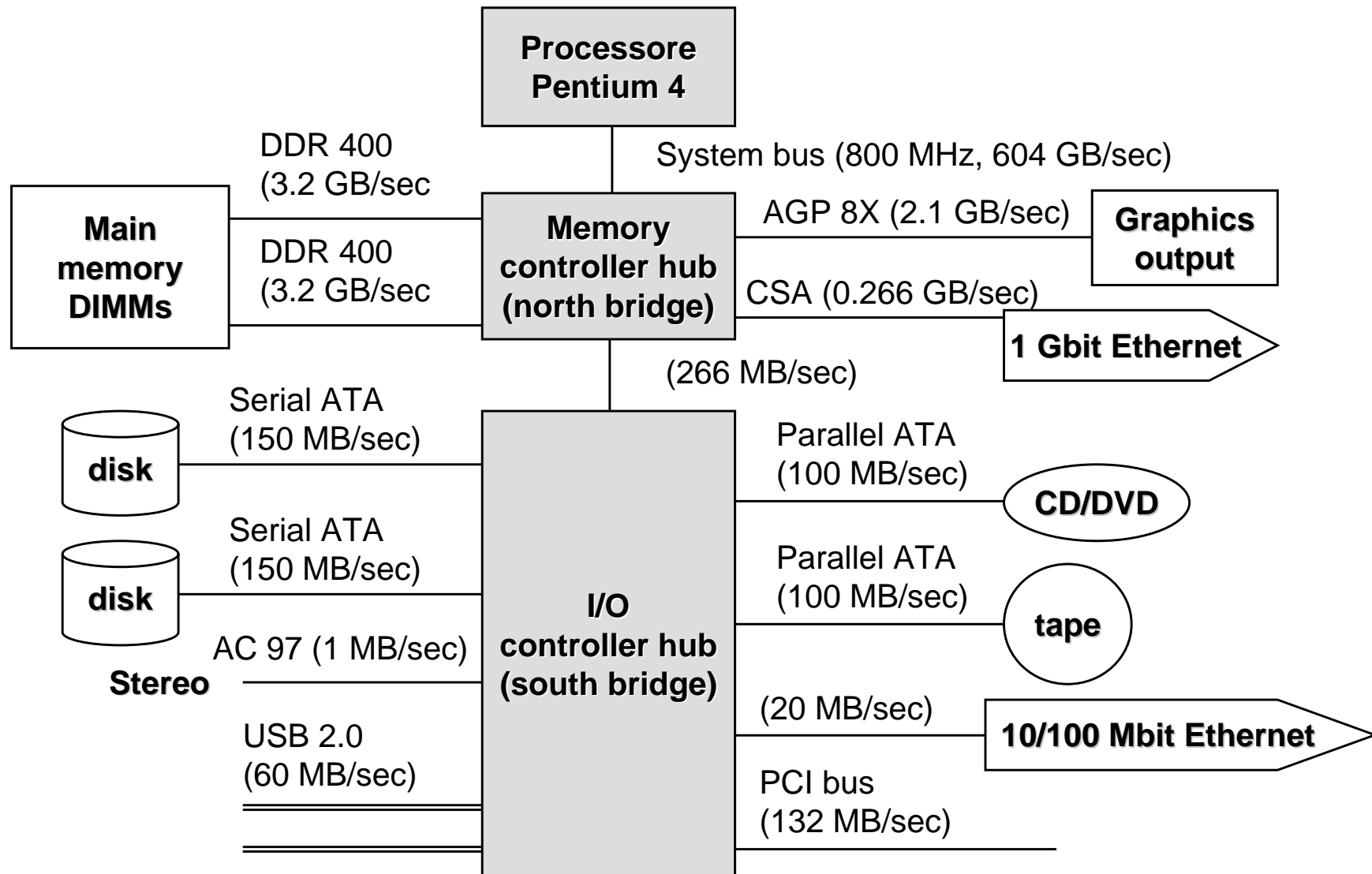
- *parallelo*
- *seriale*

bus (3)

- ISA
- Micro channel
- EISA
- VESA
- PCI
- AGP
- USB
- IDE
- Serial ATA
- PCMCIA
- SCSI
- Firewire
- Bluetooth

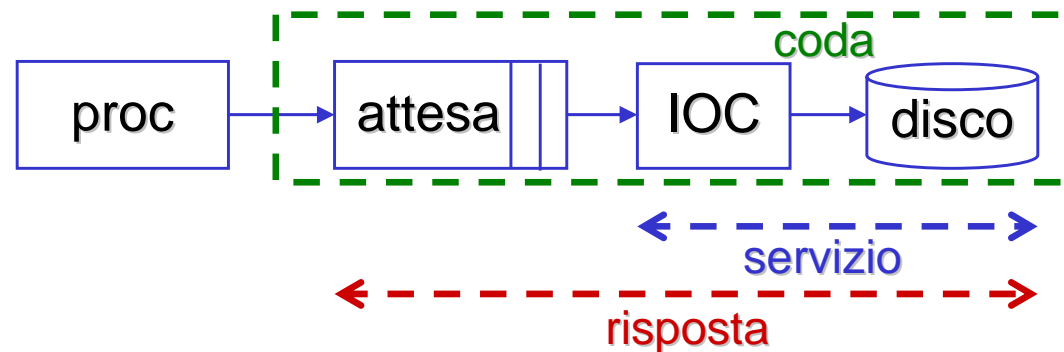
<i>Key characteristics of two dominant I/O bus standards</i>		
Characteristic	Firewire (1394)	USB 2.0
Bus type	I/O	I/O
Basic data bus width (signals)	4	2
Clocking	asynchronous	asynchronous
Theoretical peak bandwidth	50 MB/sec (Firewire 400) 100 MB/sec (Firewire 800)	0.2 MB/sec (low speed) 1.5 MB/sec (full speed) 60 MB/sec (high speed)
Hot plugable	yes	yes
Maximum number of devices	63	127
Maximum bus length (copper wire)	4.5 meters	5 meters
Standard name	IEEE 1394, 1394b	USE Implementors Forum

organizzazione I/O in Pentium 4



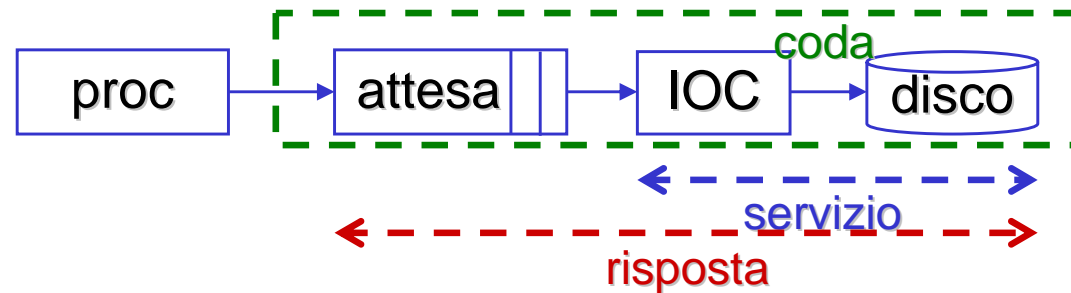
prestazioni

metriche di prestazione di un disco



- **tempo di servizio** S_{disk} : *seek time+rotational latency+data transfer time+overhead controller*
 - tempo di **seek**: posizionamento testine (\approx ms)
 - tempo di **latenza**: tempo richiesto affinché il settore passi sotto le testine (\approx ms, $\frac{1}{2}$ giro)
 - tempo di **trasferimento**: tempo di trasferimento dei dati, dipende da velocità di rotazione, densità di registrazione, distanza della testina dal centro del disco (\approx MB/sec)
 - **overhead controller**: gestione trasferimento dati da buffer locale e invio interrupt

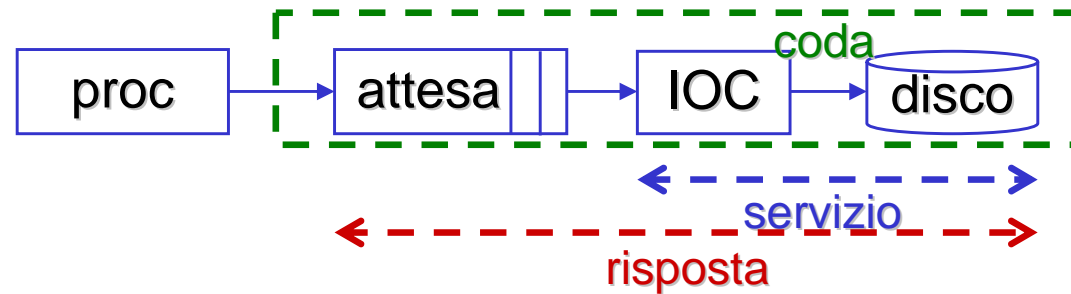
metriche di prestazione di un disco (2)



- il *tempo di risposta* r comprende anche il tempo di attesa causato dalla contesa con altre operazioni concorrenti
- dipende da:
 - numero di utenti trovati in attesa (lunghezza della coda)
 - utilizzo del "servente"
 - tempo medio di servizio
 - variabilità del tempo di servizio (forma della sua distribuzione)
 - tasso di arrivi e sua distribuzione
 - *a parità delle altre condizioni una maggiore variabilità determina un tempo di risposta mediamente più grande*

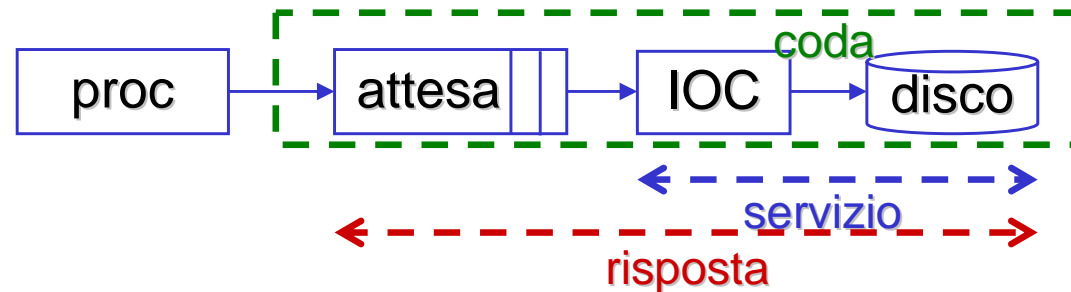
metriche di prestazione di un disco (3)

Calcolo delle medie
risultanti da composizione
o somma di diversi fenomeni



- *il tempo di servizio T_i* varia da richiesta a richiesta:
 - *Tempo medio* $t_i = M1 = (f1 \times T1 + f2 \times T2 + \dots + fn \times Tn)$
($f1 + f2 + \dots + fn = 1$)
 - *Varianza* $V_i = (f1 \times T1^2 + f2 \times T2^2 + \dots + fn \times Tn^2) - M1^2$
 - C^2 = quadrato del coefficiente di variazione
 - $C^2 = \text{Varianza} / M1^2$
- *un tempo T* è il risultato della somma di tempi indipendenti:
 - $T = t1 + t2 + \dots + tm$ (seek + rotazione + trasferimento + overhead)
 - $V = V1 + V2 + \dots + Vm$

metriche di prestazione di un disco (4)



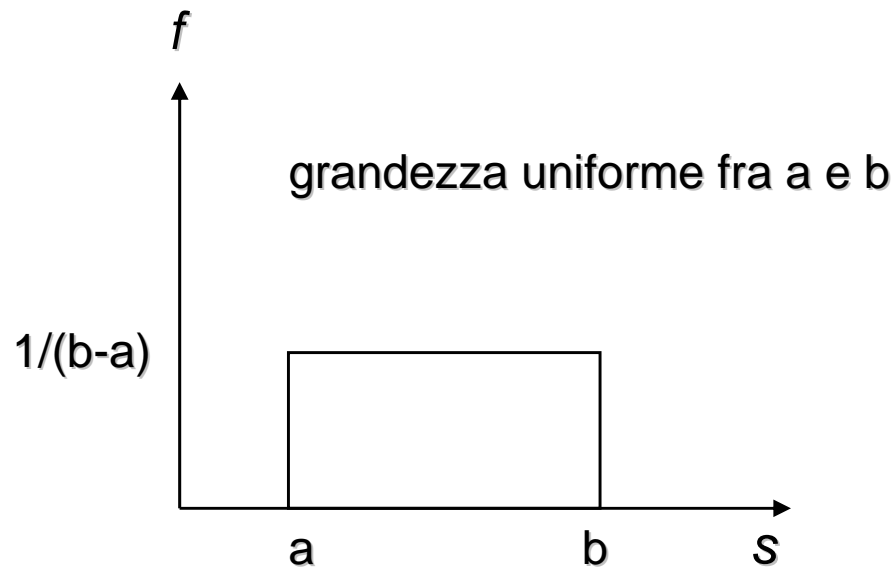
- il *coefficiente di variazione* $C = \text{stdDev}/M1$ è un indice normalizzato della variabilità della distribuzione
- C cresce all'aumentare della dispersione
- $C = 1$ (esponenziale)
 - 63% dei valori $< M1$; 90% dei valori $< 2.3 M1$
- $C < 1$ (ipoesponenziale); $C > 1$ (iperesponenziale)
- $C^2 = 0.5$
 - 57% dei valori $< M1$; 90% dei valori $< 2 M1$
- $C^2 = 2$
 - 69% dei valori $< M1$; 90% dei valori $< 2.8 M1$

tempo di servizio di una op. di I/O (esempio di calcolo)

- lettura/scrittura di un **settore** di 512 Byte = 0.5 KB,
 - velocità di **rotazione**: 10000 RPM,
 - velocità di **trasferimento** dati: 50 MB/sec,
 - **seek** medio: 6ms,
 - overhead **controller**: 0.2ms
 - **latenza**: $(60\text{s/min}) \times 1000 / (2 \times 10000 \text{ giri/min}) = 3.0\text{ms}$
 - **trasferimento**: $(512\text{B}) \times 1000 / (50\text{MB/s}) = 0.01\text{ms}$
- tempo totale di I/O = $6\text{ms} + 3\text{ms} + 0.01\text{ms} + 0.2\text{ms} = 9.21\text{ms}$
 - **località dei dati**: seek solo nel **25%** delle operazioni
$$(6.0 \times 0.25) + (0.5 \times 60 \times 10^3 / 10000) + (0.5 \text{ KB} / 50\text{MB} \times 10^3) + 0.2 =$$
$$1.5 + 3.0 + 0.01 + 0.2 = 4.71 \text{ ms}$$
- **s = seek + latenza + trasferimento + controller**

tempo di servizio di una op. di I/O (esempio di calcolo) (2)

- seek: uniforme fra 0.4 e 11.6 ms (avviene nel 25% delle operazioni) - media = 6 ms.
- latenza di rotazione: uniforme fra 0 e 6 ms - media = 3 ms.
- trasferimento: costante di 0.01 ms
- controller: costante di 0.2 ms
- *tempo medio* = $0.25 \times 6 + 3 + 0.01 + 0.2 = 4.71$ ms



$$\text{media } M1 = (b+a) / 2$$

$$\text{2do Momento } M2 = (b^2 + ab + a^2) / 3$$

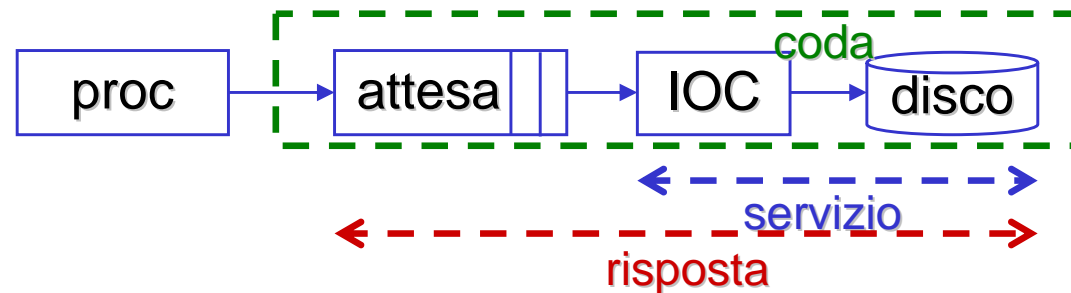
$$V = M2 - M1^2$$

$$\text{Varianza } V = (b-a)^2 / 12$$

Varianza del tempo di servizio (esempio di calcolo)

- V(tempo di seek):
 - $M2 = 0.25 \times (11.6^2 + 11.6 \times 0.4 + 0.4^2) / 3 = 0.25 \times 46.45$
 - $V = M2 - M1^2 = 11.61 - (0.25 \times 6)^2 = 9.36$
- V(tempo di latenza):
 - $V = 6^2 / 12 = 3$
- V(tempo di trasferimento):
 - $V = 0$
- V(tempo di controller):
 - $V = 0$
- *Varianza totale* = $9.36 + 3 + 0 + 0 = 12.36$
- *coeff. di variazione* = $\sqrt{(12.36/4.7)} = 0.748$
- $0.748 < 1 \Rightarrow$ distribuzione ipoesponenziale

metriche di prestazione di un disco (5)

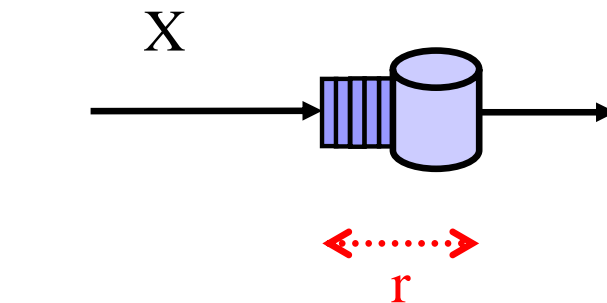


- in condizioni di equilibrio *tasso di richieste in arrivo = tasso di partenza = X*
- s tempo medio di servizio
- u utilizzo ($u = s \times X$)
- w tempo medio di attesa
- r tempo medio di risposta ($r = s + w$)
- L_w numero medio di richieste in attesa
- L_q numero medio di richieste in coda ($L_q = u + L_w$)
- *legge di Little* $X = L_q/r = L_w/w = u/s$

Modello M/M/1 (soluzione intuitiva)

- sistema in equilibrio
- s : tempo di servizio
- X : flusso di operazioni
- L_q : lunghezza della coda
- u : utilizzo = $X \times s$
- r : tempo di risposta
- $L_q = X r$ (legge di Little)
- $r = s + w = s + s L_q$
- w = (tempo di attesa)
- $r = s + s X r$

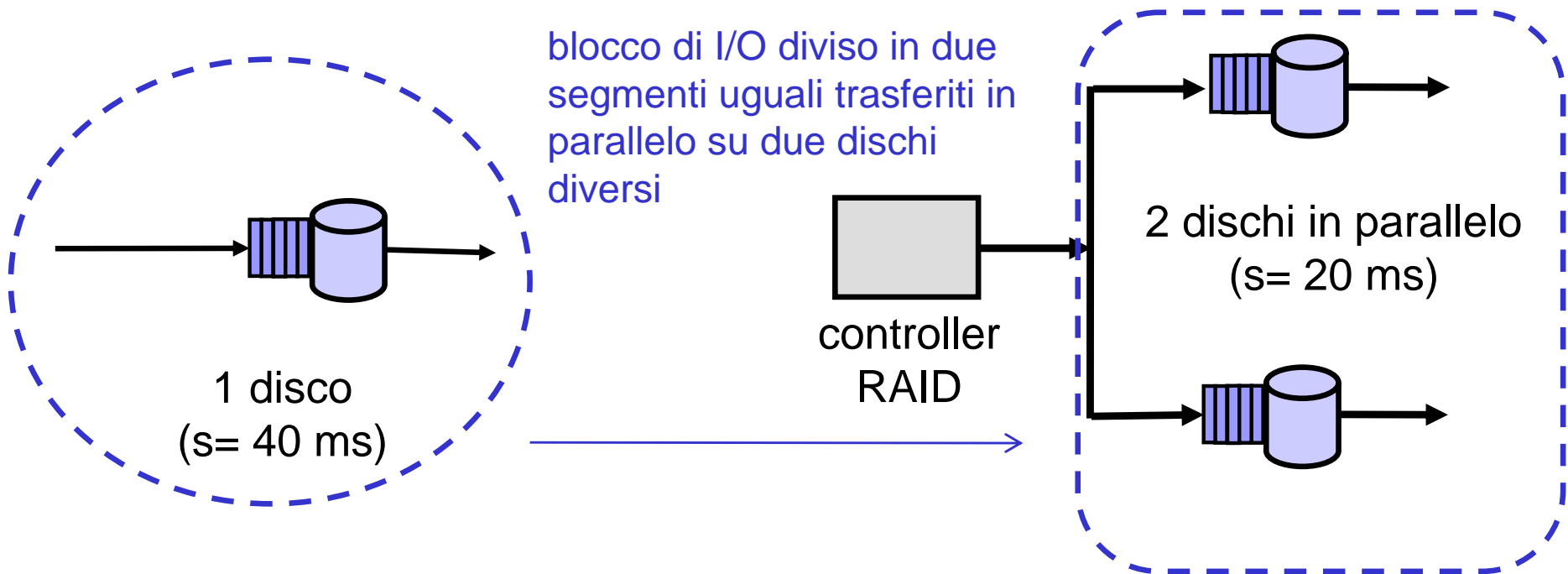
- $r = s / (1 - X s) = s / (1 - u)$



Valori medi

- Il modello M/M/1 è il punto di partenza per lo studio delle prestazioni degli impianti informatici

un esempio didattico: 2 dischi in parallelo

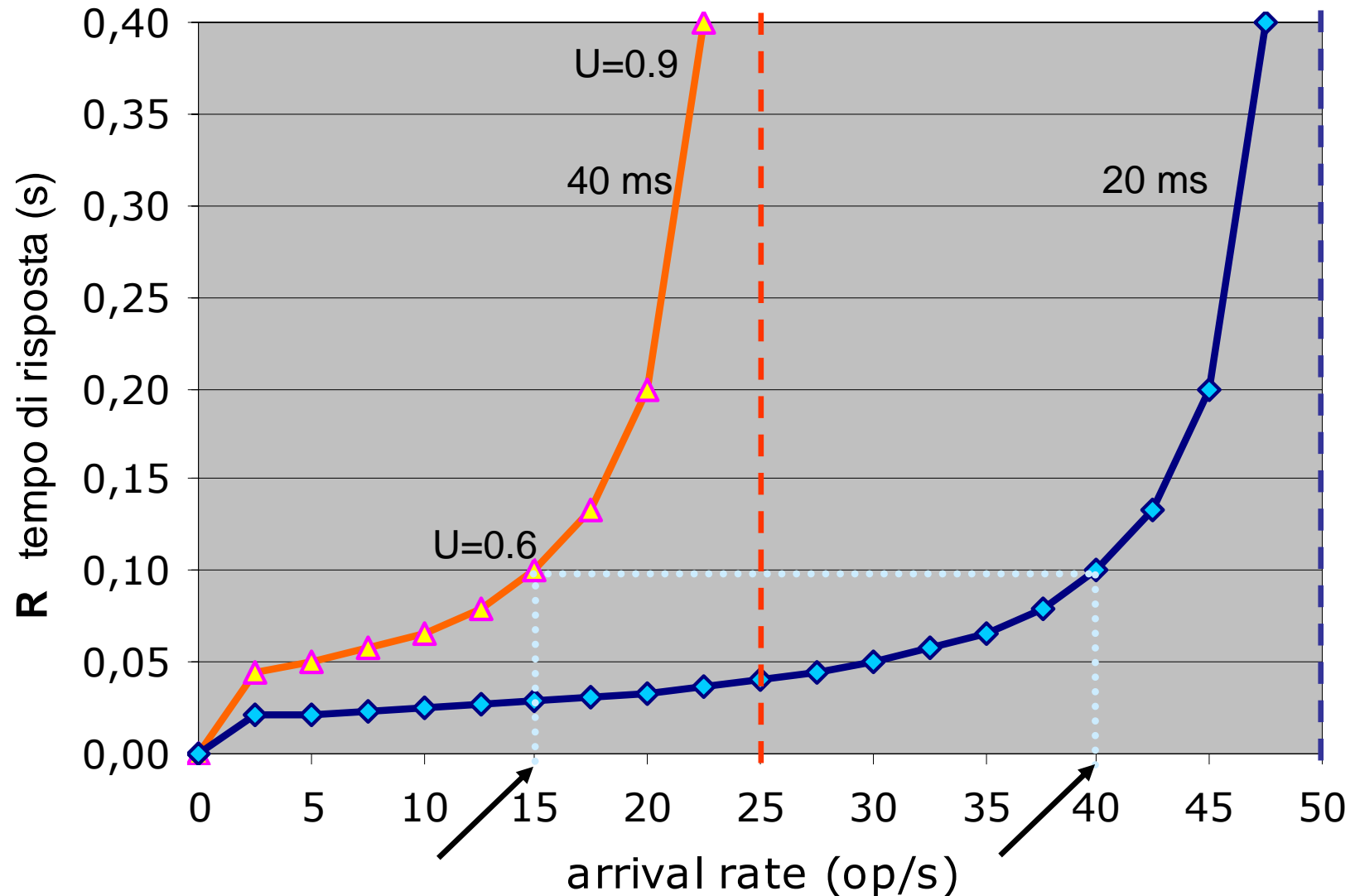


in prima approssimazione la situazione può essere studiata ipotizzando che le operazioni vengano effettuate in parallelo sui due dischi caratterizzati da tempi di servizio pari alla metà di quello originario ($s=20$ ms)

si noti, ad esempio che, a **parità di utilizzo** passando da uno a due dischi in parallelo, il tempo richiesto da una singola operazione di I/O si dimezza e il carico si raddoppia (utilizzo 60%, tempi di risposta da 100 ms a 50 ms). A **parità di tempo di risposta** il carico di lavoro servito cresce (per esempio aumenta del 166% da 15 a 40 operazioni/s con 100 ms di risposta)

esempio didattico: calcolo del tempo di risposta

Andamento dei tempi di risposta al variare del carico (operazioni/sec)
per tempi di servizio di 40 e 20 ms



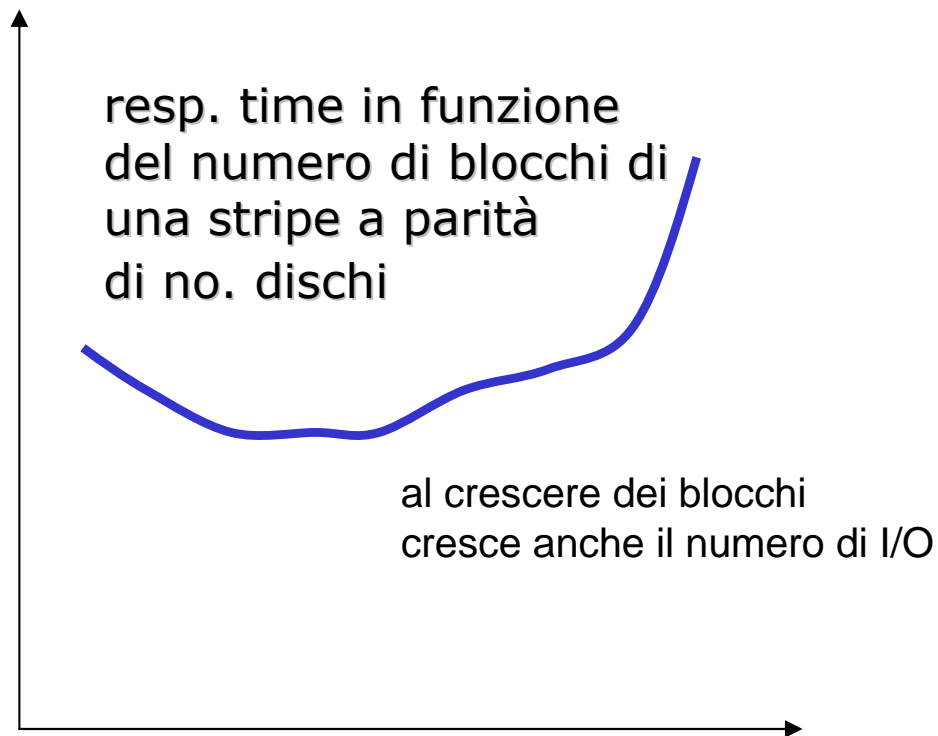
tecniche per migliorare le prestazioni di I/O

tecniche per migliorare le prestazioni I/O

- le prestazioni I/O possono essere misurate sperimentalmente a diversi livelli della gerarchia di storage.
 - per quantificare sperimentalmente gli effetti delle diverse tecniche di ottimizzazione bisogna misurare i tempi da quando la richiesta è consegnata al sistema storage prima che venga potenzialmente spezzata dal *volume manager* in richieste dirette a dischi multipli.
- due metriche importanti sono *response time* e *throughput*.
- il reciproco del *service time* è una stima (ottimistica) del *throughput massimo* (ottenibile con utilizzo = 1)

locality su dischi multipli

- Quando i dati sono suddivisi su più dischi (*striping*) la *locality* è modificata: le regioni attive possono essere contigue su più dischi così che i bracci di posizionamento delle testine hanno una estensione di movimento più limitata, infatti lo stesso carico utilizza su ognuno dei diversi dischi solo una frazione dello spazio che userebbe su uno.

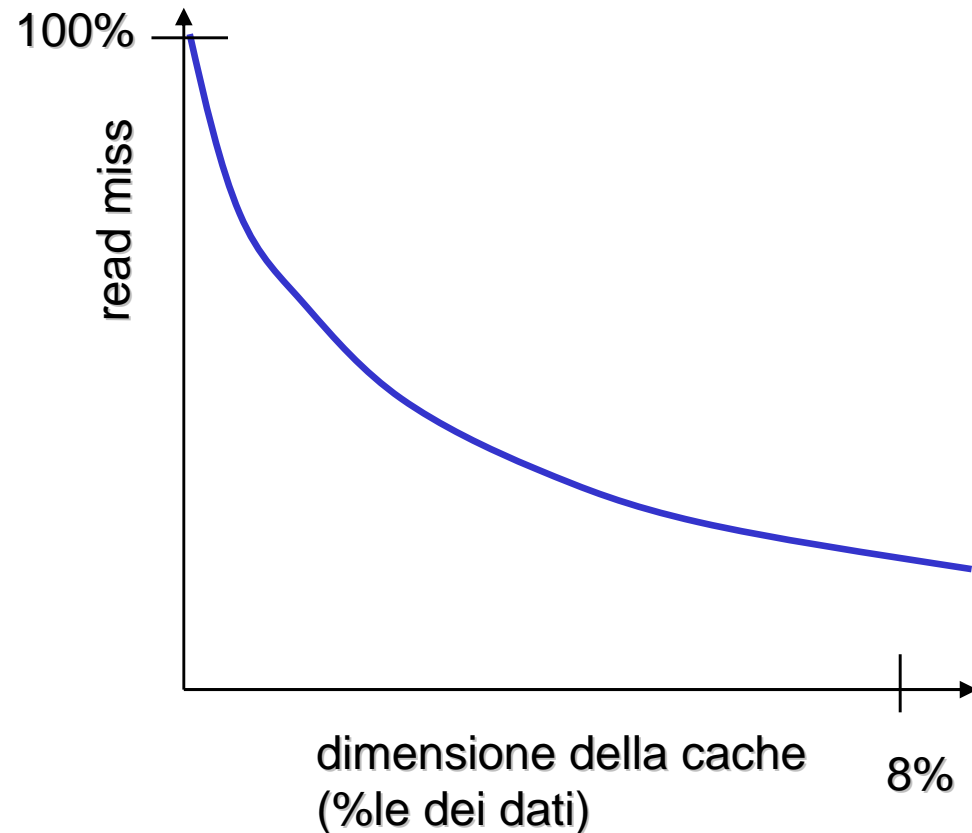


read caching

- tecnica che consiste nel tenere temporaneamente in una memoria più veloce (*cache*) quei dati che si pensa sia probabile vengano usati in seguito.
 - dati: blocchi di storage (generalmente di 4KB);
 - cache: DRAM (efficiente se > 32MB) in memoria;
 - anche molti dischi hanno una loro unità cache (2-4-8 MB) che serve soprattutto come buffer di *prefetching*;
 - metodo di alimentazione *least-recently-used* (LRU).
- *read miss ratio*: frazione di operazioni che richiedono l'operazione fisica sul disco
 - continua a migliorare al crescere delle dimensioni della cache almeno fino a che essa raggiunge il 4% della memoria storage usata

read caching (2)

- l'analisi di dati sperimentali mostra una *miss ratio* che segue la relazione:
 - $f(x) = a(x-b)^c$ (a, b, c costanti, $c \cong -1$)
- la relazione funzionale è simile a quella che si trova a livello logico per i buffer dei database (ma in questo caso il valore di c è circa la metà, in altre parole si trova che il *caching* a livello fisico è più efficiente di quello ottenuto a livello logico).



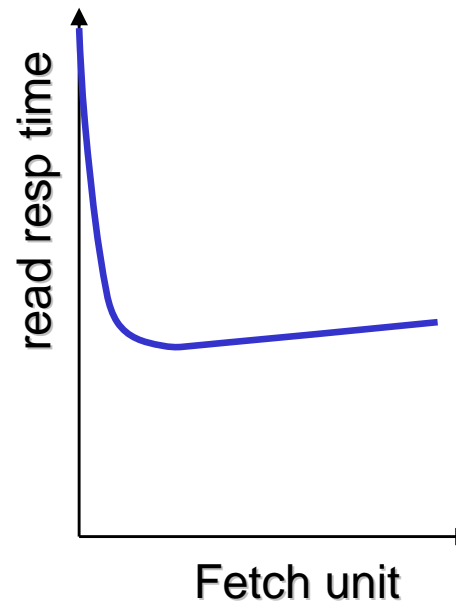
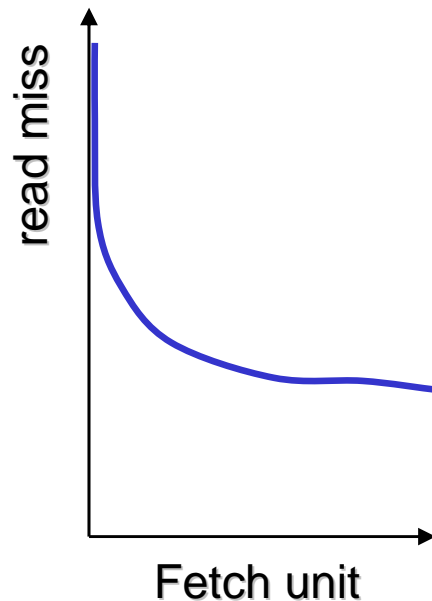
prefetching

- consiste nel prevedere quali blocchi è probabile vengano usati in futuro e caricarli prima che siano effettivamente richiesti, la sua efficienza dipende da:
 - accuratezza della previsione;
 - costo: risorse consumate (memoria e altri componenti);
 - tempestività (perché l'operazione deve essere completata prima che i blocchi siano richiesti).
- la maggior parte dei carichi presenta una certa sequenzialità nella distribuzione degli accessi
 - un *prefetch* sequenziale eseguito al cache miss trasforma in un singola operazione parecchie I/O di blocchi più piccoli

prefetching (2)

■ *Large fetch unit*

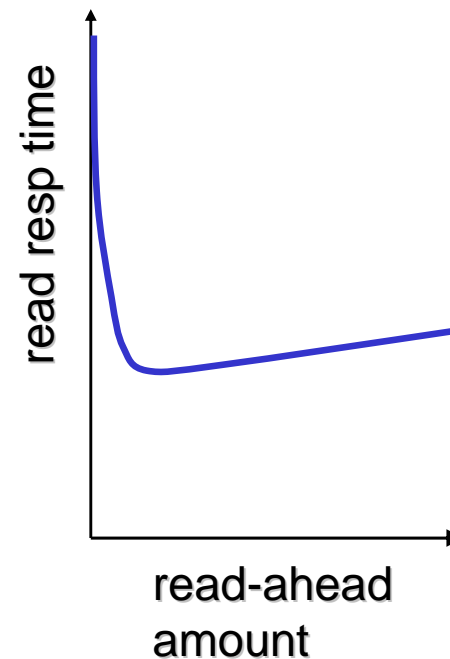
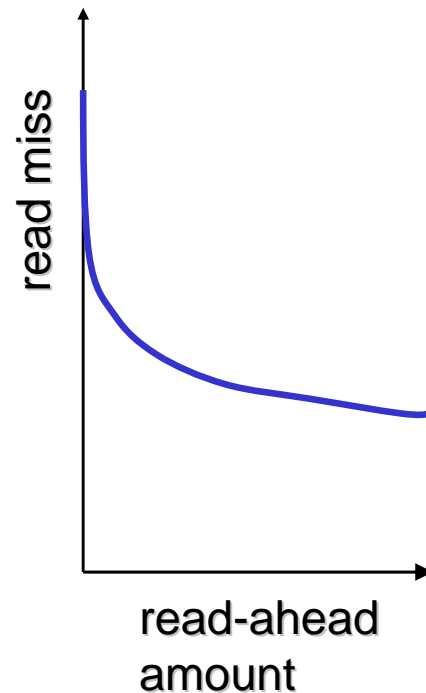
- tecnica che esegue il caricamento di un certo numero di blocchi che precedono e che seguono quello effettivamente richiesto;
- *response time penalty* bisogna attendere che il trasferimento sia completato, in alternativa si possono eseguire due operazioni: fino al blocco "target" e per i blocchi rimanenti



prefetching (3)

- *Read ahead*

- dopo che i blocchi richiesti sono stati ottenuti, si prosegue a leggere in avanti, generalmente vengono eseguite due operazioni: una per i blocchi “target” e un'altra per quelli che seguono



prefetching (4)

Una semplice forma di *prefetch* che usa solo risorse che sarebbero altrimenti *idle* o non usate è:

- *Preemptible read ahead*
 - la richiesta è divisa in tante sotto-richieste, tale sequenza è interrotta quando arriva una nuova richiesta. In questo modo si evita che al crescere della domanda (numero di blocchi) le prestazioni comincino a degradare.
 - Il metodo migliore è un approccio ibrido: si inizia a leggere la traccia su cui si è posizionati, si prosegue di 32 KB oltre i blocchi richiesti e, se non ci sono nuove richieste, si prosegue fino a 128 KB. I miglioramenti di prestazione, nel caso dei server sono almeno del 50% rispetto ai sistemi senza prefetching.

write buffering

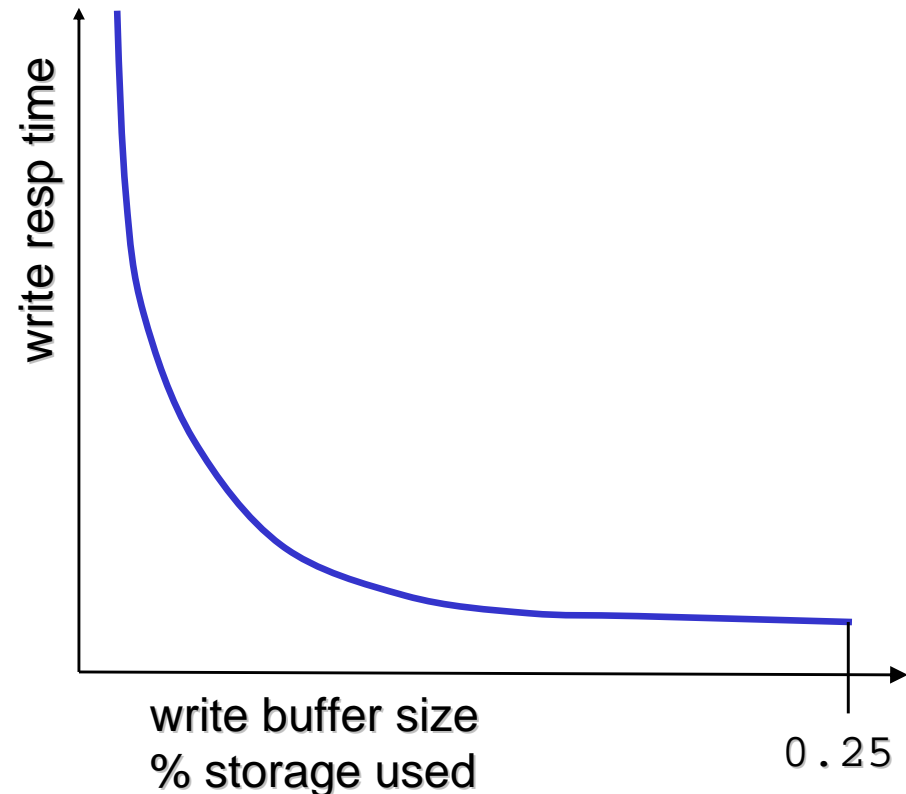
- consiste nel mantenere temporaneamente in memoria i blocchi scritti prima di fare un *destaging* dei dati nella storage permanente
 - l'operazione di write è data come completata quando il dato è scritto nel buffer;
 - (il tempo di latenza è nascosto e differito)
 - per evitare perdita di dati il *write buffer* è eseguito su memoria non volatile (NVS) o, in un approccio più economico, periodicamente (es. ogni 30 secondi) il contenuto dei buffer trasferito su disco;
 - una singola write fisica combina operazioni multiple sulla stessa posizione;
 - (si riduce il numero di operazioni fisiche)
 - write multiple consecutive sono combinate in una operazione;
 - (maggiore efficienza)

write buffering (2)

- criteri: si tenta di ridurre il numero di write fisiche operando il *destage* dei blocchi su cui è meno probabile avvengano in futuro riscritture
 - il processo di *destage* del buffer inizia quando il numero di blocchi modificati è superiore a una soglia (*high mark*) e termina quando scende sotto il limite (*low mark*);
 - il blocco da scaricare è selezionato in base al metodo LRW (*least-recently-written*) o quando ha superato un massimo stabilito di età;
 - è selezionata per il *destage* la traccia col maggior numero di blocchi modificati;
 - l'eliminazione delle write ripetute e la scrittura multipla competono per spazio - l'equilibrio è raggiunto con un appropriato aggiustamento dei valori di soglia, una buona efficienza si consegue con frazioni di soglia: *high mark* = 0.8 e *low mark* = 0.2

write buffering (3)

- se la soglia minima è considerevolmente inferiore a quella massima, le operazioni di *destage* avvengono in lotti;
- le operazioni fisiche di write possono essere ordinate in modo da minimizzare il tempo di attesa selezionando le richieste da servire in base al minimo tempo di accesso (seek + latency) o al minimo tempo di posizionamento stimato (con un fattore di correzione che dipende dal tempo già trascorso in attesa)



dischi RAID

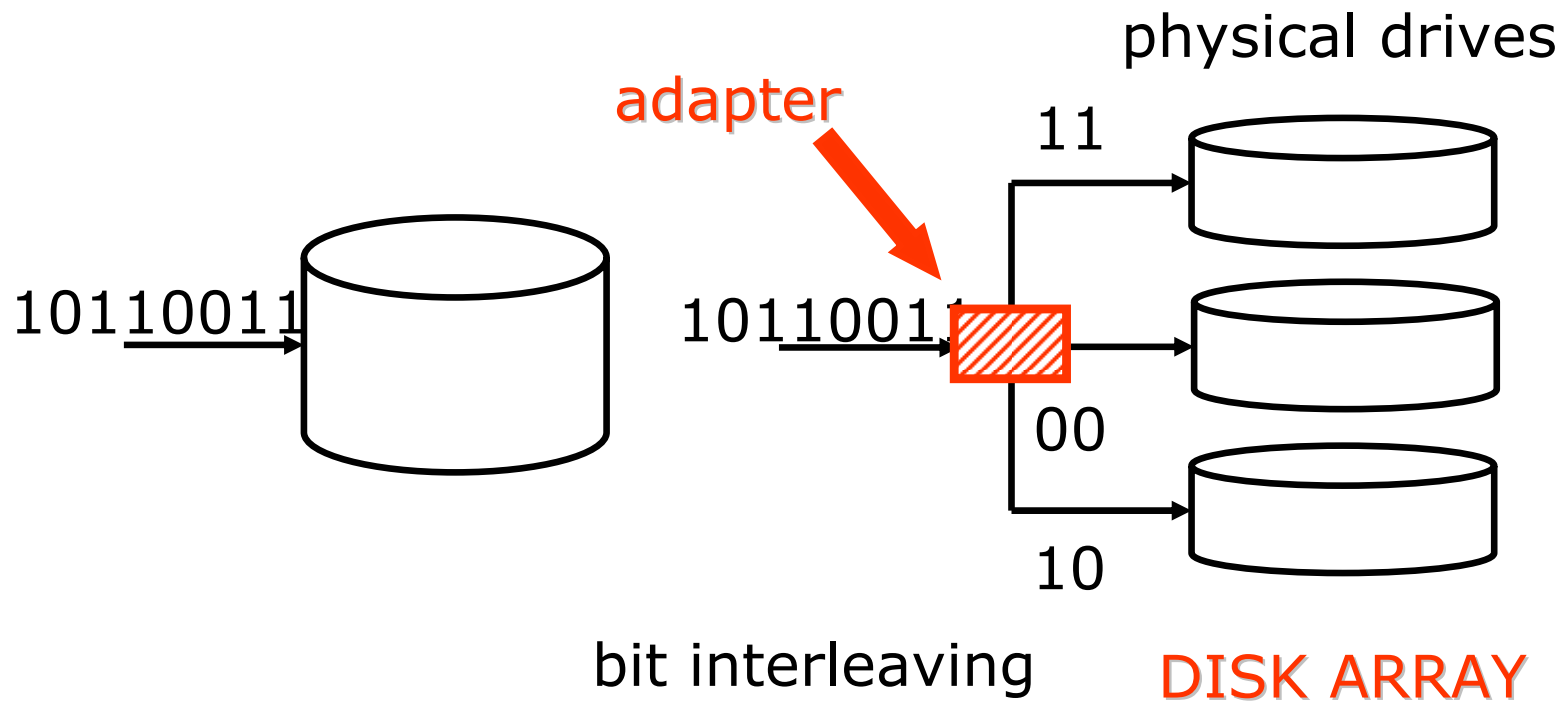
Redundant Arrays of Independent (Inexpensive) Disks

- più dischi *indipendenti* visti come un *unico grande disco logico* ad elevate prestazioni
- i dati sono distribuiti (*striped*) su più dischi che sono acceduti in *parallelo* ottenendo:
 - **elevato transfer rate** per accessi a grandi quantità di dati (operazioni di I/O pesanti)
 - **elevato I/O rate** per accessi a piccole quantità di dati (operazioni di I/O leggere)
 - **load balancing** tra i vari dischi in modo automatico
- applicano due tecniche antitetiche
- **striping** dei dati: per aumentare le **prestazioni**
- **ridondanza**: per aumentare l'**affidabilità**

Redundant Arrays of Independent (Inexpensive) Disks (2)

- possono essere realizzati sia con hardware dedicato sia con software che usa hardware standard (esistono anche soluzioni ibride)
- *soluzioni software* richiedono costi di CPU (cicli aggiuntivi)
- *soluzioni hardware*
 - richiedono unità di controllo speciali che eseguono i calcoli di parità
 - hanno in genere velocità maggiore che dipende dalla dimensione della cache e da quanto rapidamente i dati vengono scaricati sui dischi
 - supportano *hot swapping* (i dischi possono essere sostituiti anche in modo automatico mentre il sistema è in esercizio)

parallelismo di una operazione di I/O



Aumento delle prestazioni

data striping

- i dati sono distribuiti, in modo *trasparente all'utente*, su più dischi *visti* come un unico disco veloce di grande capacità
- **striping**: suddivisione di dati che devono apparire come scritti sequenzialmente (un vettore, un file, ...) in segmenti (unità di stripe) che vengono scritti su più dischi fisici con un algoritmo ciclico (round robin)
- **unità di stripe**: quantità di dati (**dimensione**) che vengono scritti su un solo disco ($\sim 2 \div 128\text{KB}$)
- **stripe width**: numero di dischi usati dall'algoritmo di striping (non coincide necessariamente col numero di dischi fisici nell'array)

data striping (2)

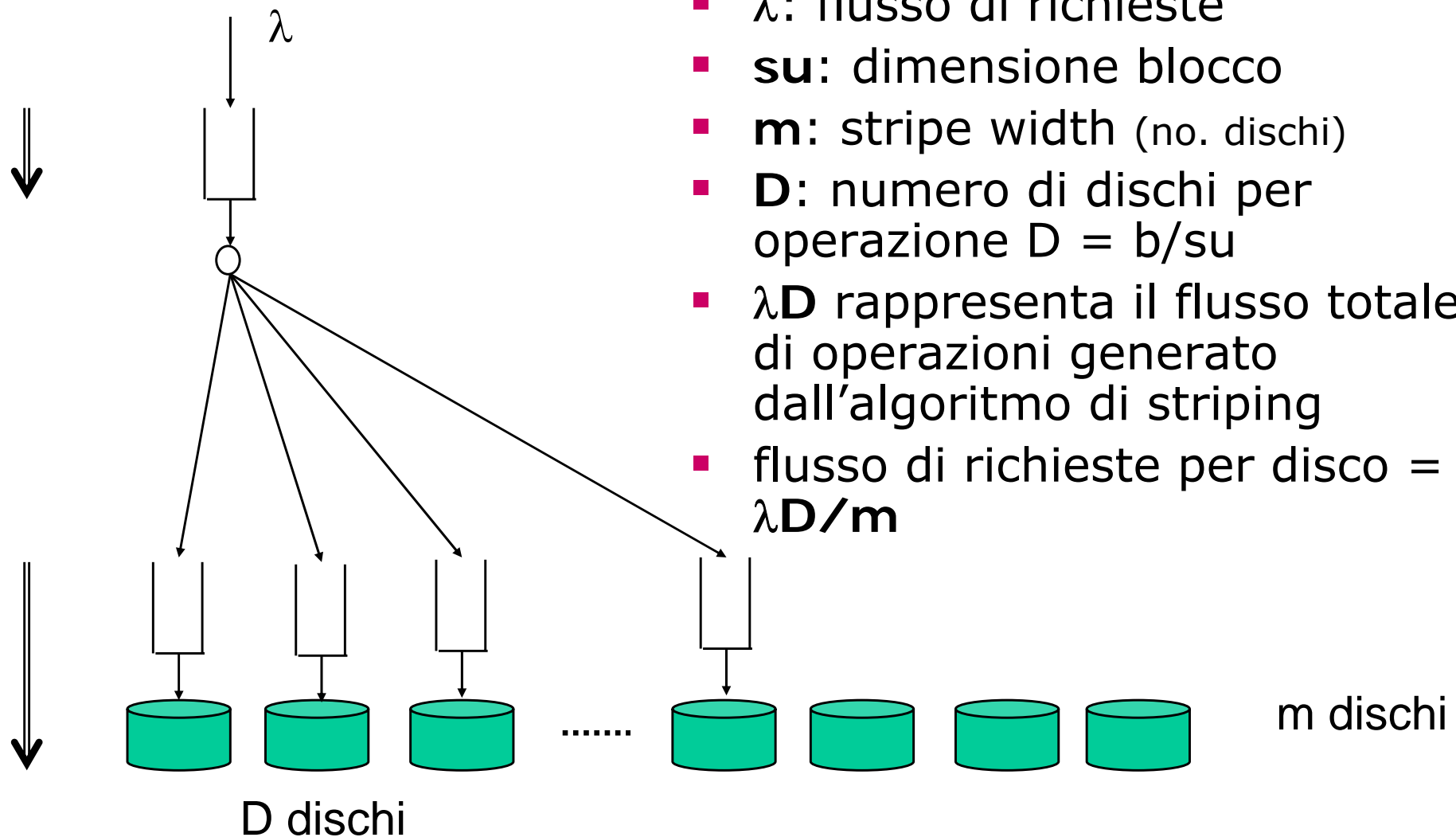
le prestazioni globali di I/O migliorano poiché più operazioni di I/O vengono eseguite in parallelo

- più richieste indipendenti vengono eseguite in parallelo da dischi diversi: si riduce il tempo di accodamento delle richieste di I/O ai dischi
- una singola richiesta di I/O per blocchi multipli può essere servita da più dischi che operano in modo coordinato: si aumenta il *transfer rate* per richiesta

data striping (3)

- la scelta della dimensione dell'unità di *stripe* è un punto importante per le prestazioni:
 - una unità piccola può dare luogo a singole richieste che si distribuiscono su più dischi con un aumento del numero di operazioni di I/O e di dischi occupati, il *prefetching* è meno efficace dal momento che dati che sarebbero contigui su un disco si trovano fisicamente dispersi e mescolati con quelli di altri dischi;
 - d'altronde così diminuisce la probabilità di avere un sottoinsieme di dischi con utilizzo sproporzionatamente asimmetrico (*access skew*)

modello semplificato di striping



ridondanza: effetto sulle prestazioni

le write devono aggiornare anche le informazioni ridondanti, quindi sono più lente delle tipiche operazioni di write

aumentata vulnerabilità



scrittura di informazioni ridondanti



lentezza delle write

aumentata vulnerabilità

- array di 100 dischi: probabilità di guastarsi superiore di 100 volte quella di un disco solo
- se un disco ha un MTTF (Mean Time To Failure) di 200000 ore (~23 anni) un array di 100 dischi avrà un MTTF di 2000 ore (~ 3 mesi)
- **ridondanza** dei dati scritti: possibilità di correzione di eventuali errori/perdite di dati (disco guasto) con tecniche di codici a correzione di errore che utilizzano informazioni ridondanti scritte su dischi diversi da quelli sui quali vengono scritti i dati

parallelismo e affidabilità (reliability)

più elevato è il numero di dischi dell'array, **maggiori** sono i benefici prestazionali ma **minore** è l'affidabilità dell'intero disk array (cioè **maggiore** è la probabilità di guasti)



per questo motivo viene introdotta la
ridondanza

tipi di RAID

la maggior parte dei RAID può essere classificata attraverso due caratteristiche:

- la **granularità** di *interleaving* dei dati (unità di stripe)
- il **metodo** di calcolo dei dati ridondanti ed il **modo** col quale sono distribuiti tra i vari dischi

data interleaving: fine grained

- i dati vengono interleaved in piccole unità in modo tale che tutte le richieste di I/O (indipendentemente dalle loro dimensioni) possono utilizzare tutti i dischi dell'array
 - + elevato transfer rate per tutte le richieste di I/O
 - una sola richiesta logica di I/O può essere servita ad ogni istante di tempo (perché vengono utilizzati molti dischi)
 - tutti i dischi possono perdere parecchio tempo per motivi di posizionamento ad ogni richiesta

data interleaving: coarse grained

- i dati vengono interleaved in (relativamente) grandi unità in modo tale che tutte le richieste di I/O di piccole dimensioni necessitano soltanto di pochi dischi mentre grandi richieste di I/O possono utilizzare tutti i dischi dell'array
 - + richieste di I/O *molte piccole* possono essere eseguite in modo concorrente
 - + richieste di I/O *molto grandi* possono avere elevato transfer rate in quanto usano molti dischi

due problemi ortogonali

- 1 scelta del metodo per il calcolo delle informazioni ridondanti
- 2 scelta del metodo per distribuire le informazioni ridondanti tra i dischi dell'array
 - metodi che distribuiscono le info ridondanti su un numero **limitato** di dischi
 - metodi che invece utilizzano tutti i dischi disponibili in modo **uniforme** per allocare le info ridondanti (load balancing ottimale)

esempio di ridondanza

Disco 1	Disco 2	Disco 3	dati ridondanti
10	8	2	20
10	guasto	2	20

$$\text{guasto} = 20 - (10 + 2) = 8$$

Disco 1	Disco 2	Disco 3	parità
1	1	0	0
1	guasto	0	1

$$\text{parità} = \text{somma modulo } 2$$

$$\text{guasto} = \text{parità} - (1+0) = 0$$

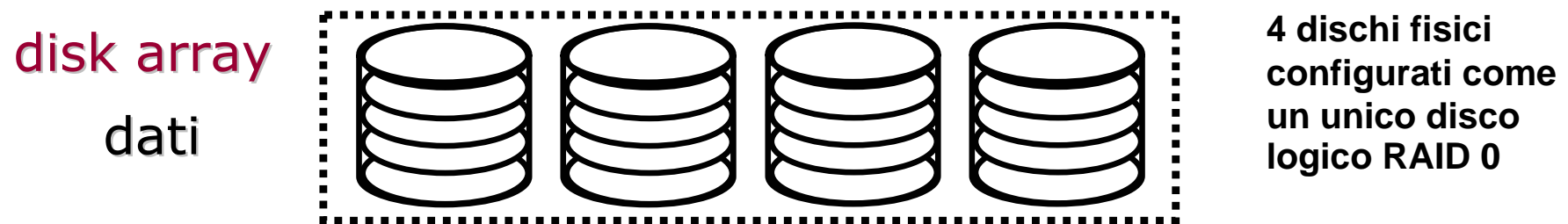
tipologie di RAID

i livelli di RAID

- RAID 0 striping
- RAID 1 mirroring
 - RAID 0+1
 - RAID 1+0 (livelli annidati)
- RAID 2 bit interleaving (non usato)
- RAID 3 byte interleaving - ridondanza (disco di parità)
- RAID 4 block interleaving - ridondanza (dischi di parità)
- RAID 5 block interleaving - ridondanza (blocchi di parità distribuiti) - **il più usato**
 - RAID 5+0 (livelli annidati)
- RAID 6 ridondanza (tollera 2 dischi guasti)
- RAID 7 (soluzioni proprietarie)

RAID level 0: *striping, no ridondanza*

- i dati scritti su un disco logico sono suddivisi in *blocchi* sequenziali e distribuiti tra i dischi fisici con un algoritmo di striping
- richiede minimo 2 drive
- + massima capacità per i dati, no ridondanza (**costi minimi**)
- + **massime prestazioni** (parallelismo di dischi e canali)
- + **massime prestazioni** di **write** (no redundant data update)
- non si possono usare dischi hot spares
- senza correzione di errori, no fault tolerant (**min. affidabilità**)



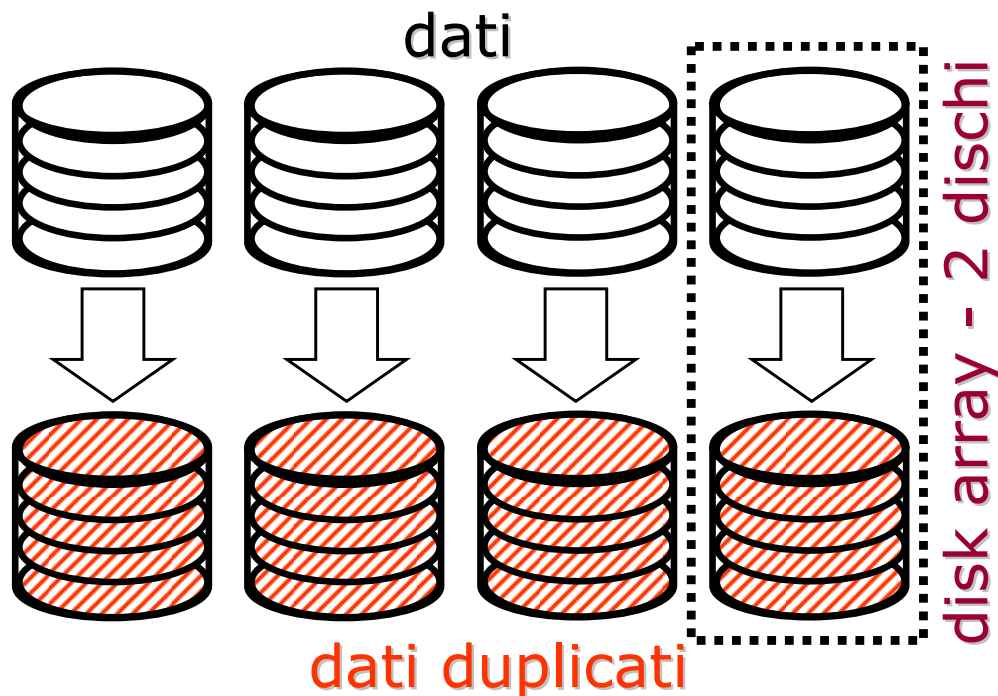
usato nei server
di dati
read-only

	Disco 1	Disco 2	Disco 3	Disco 4	
Stripe 1	<i>Block 1</i>	<i>Block 2</i>	<i>Block 3</i>	<i>Block 4</i>	
Stripe 2	<i>Block 5</i>	<i>Block 6</i>	<i>Block 7</i>	<i>Block 8</i>	

RAID level 1: *mirroring*

i dati di un disco vengono tutti **duplicati** su un altro disco

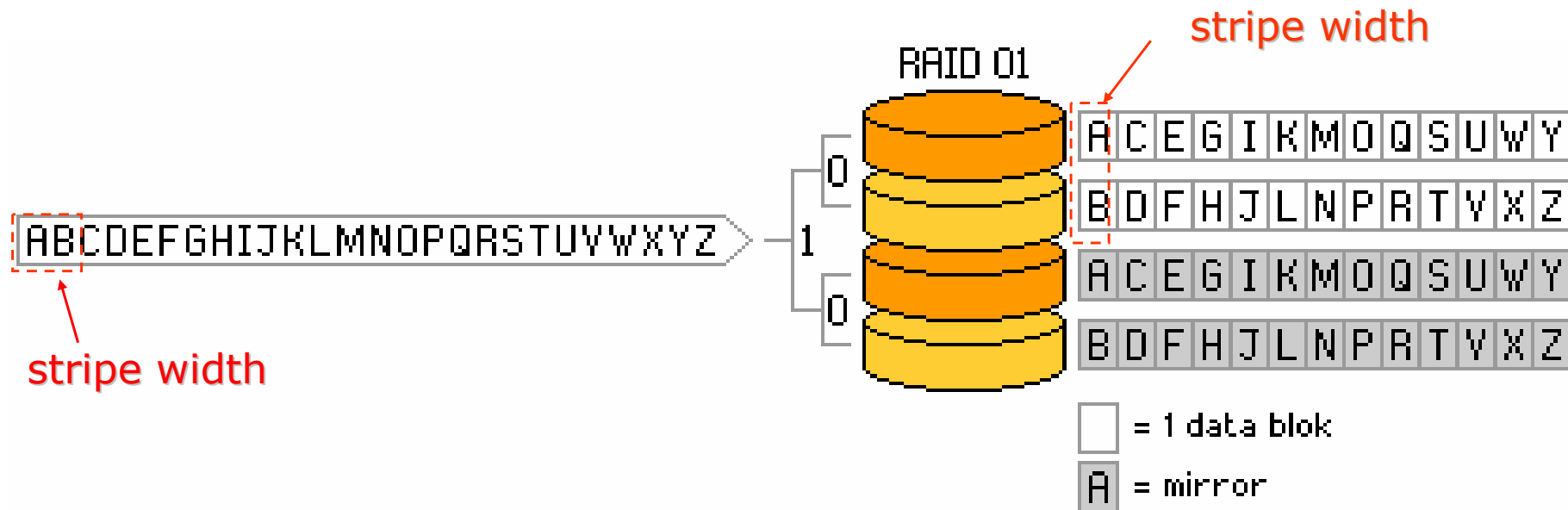
- richiede minimo 2 drive
- + alta affidabilità; un disco si guasta: si accede al duplicato
- + **read** veloci (un disco è occupato: si legge dall'altro)
- costi elevati (si utilizza solo il 50% della capacità fisica per dati utili)



	Disco 1	Disco 2
Stripe 1	<i>Block 1</i>	<i>Block 1</i>
Stripe 2	<i>Block 2</i>	<i>Block 2</i>
Stripe 3	<i>Block 3</i>	<i>Block 3</i>

RAID level 0 + 1

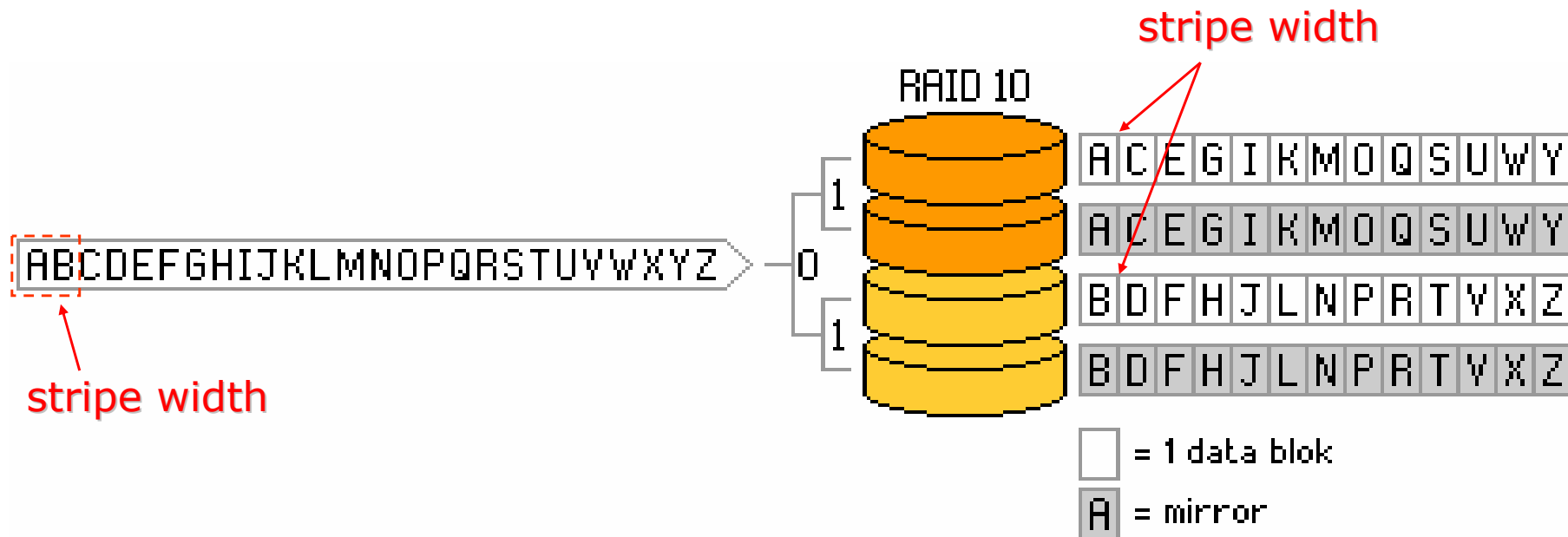
- prima viene effettuato lo striping (modello 0)
- poi viene effettuato il mirroring (modello 1)



- elevate prestazioni
- richiede un minimo di 4 drive
- il guasto di un disco porta nella situazione RAID 0
- overhead elevato (duplicazione dischi)

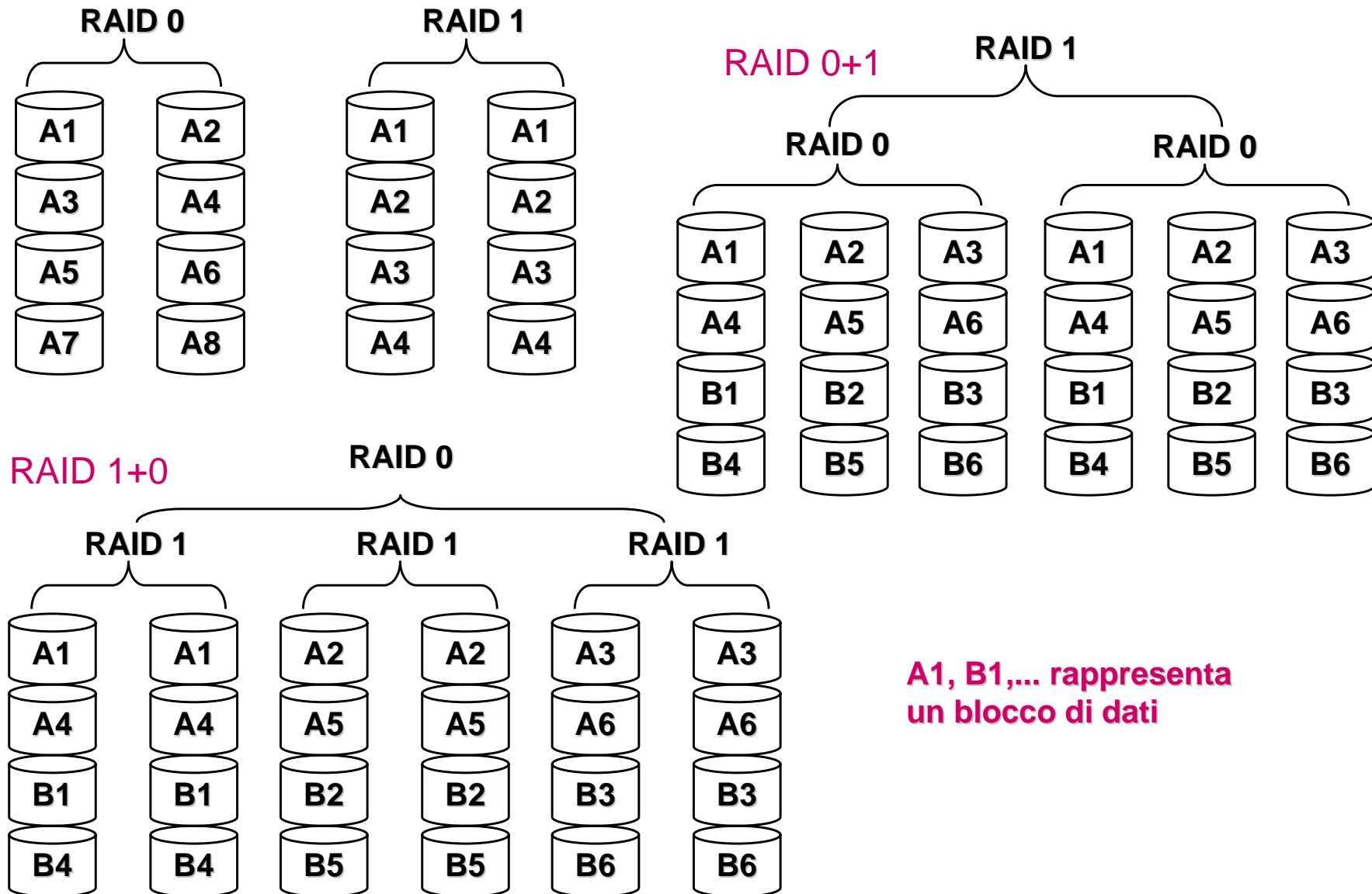
RAID level 1 + 0

- prima viene effettuato il mirroring (modello 1)
- poi viene effettuato lo striping (modello 0)



- elevata affidabilità combinata con elevate prestazioni
- richiede un minimo di 4 drive
- stessa *fault tolerance* di RAID 1
- impiegato in data base ad elevato carico (write veloci)

schema RAID 0, 1, 0+1, 1+0

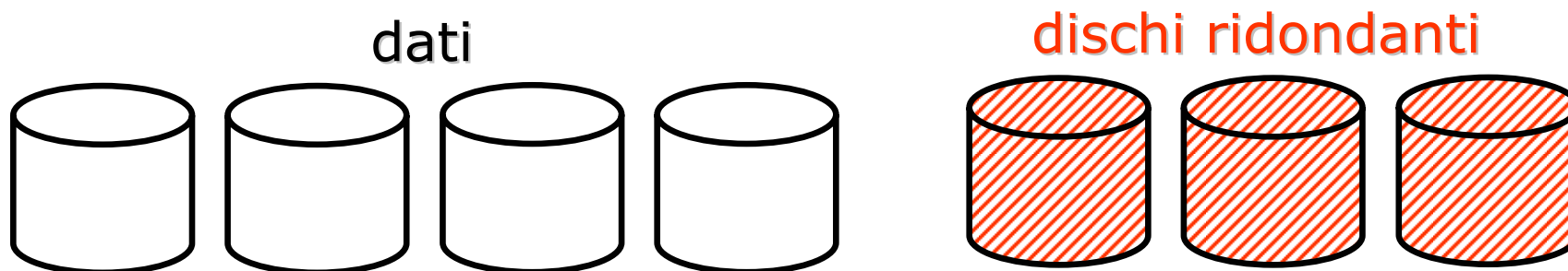


confronto livelli 0+1, 1+0

- la disposizione dei blocchi è identica se non per i dischi che sono in un diverso ordine
- alcuni controller 0+1 combinano in un'unica operazione striping e mirroring
- 0+1
 - non tollera due guasti simultanei (eccetto nel caso in cui interessino la stessa stripe)
 - nel caso di guasto a un singolo disco, qualunque guasto ad altra stripe è un *single point of failure*
 - il ripristino del disco richiede la partecipazione di tutti i dischi dell'array
- 1+0
 - un disco per ogni gruppo RAID 1 può guastarsi ma se non riparato, qualunque altro disco è *single point of failure* dell'intero array

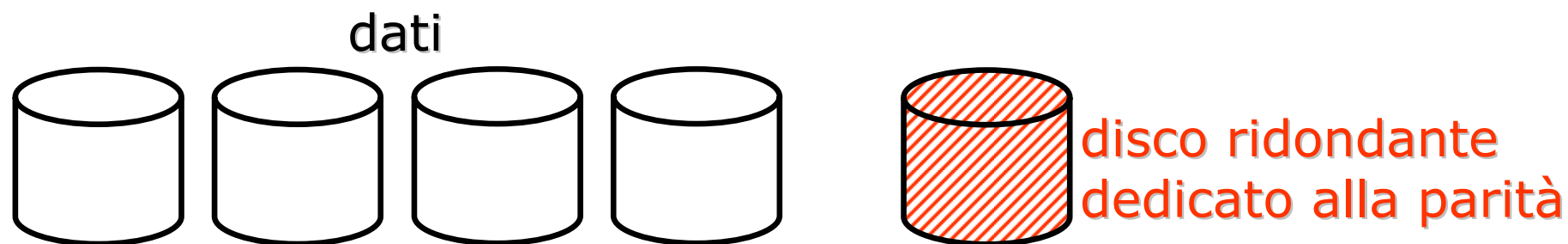
RAID level 2 *bit interleaved* parity (non usato)

- assunzione: non si conosce quale è il dato errato
- con correzione di errori (Hamming code) del tipo usato nelle memorie
- 4 dischi dati \Rightarrow 3 dischi ridondanti (un disco in meno del mirroring)
- num. dischi ridondanti: proporzionale a \log num.dischi
- si usano blocchi di parità per sottoinsiemi di dati sovrapposti
- molti dischi ridondanti sono necessari per individuare il disco guasto, un solo disco è necessario per ripristinare i dati perduti
- in lettura viene verificata la correttezza dei dati e corretti gli errori su un singolo drive



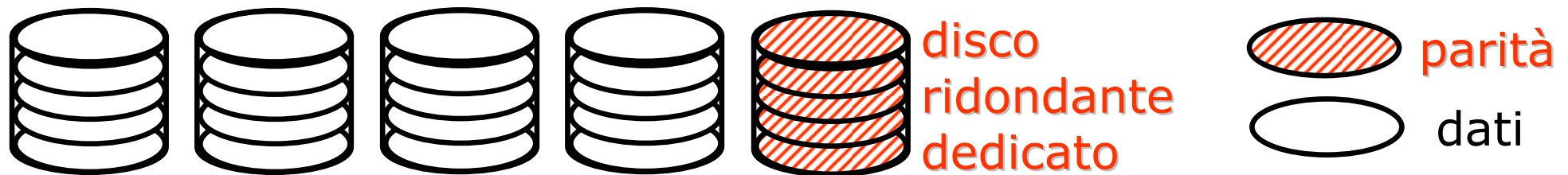
RAID level 3 *byte interleaved parity*

- assunzione: si conosce quale è il disco guasto
- i bit di un dato vengono distribuiti tra i vari dischi
- un solo disco per i bit di parità
- adatto per applicazioni che richiedono elevata banda ma medio I/O rate
- una sola richiesta di I/O viene eseguita per volta
- ogni read accede a tutti i dischi dati
- ogni write accede a tutti i dischi dati ed al disco di parità



RAID level 4: *block interleaved* parity

- i dati sono distribuiti in blocchi (non in bit)
- read inferiori ad un blocco usano un solo disco
- le write devono scrivere i nuovi dati, leggere i vecchi dati e parità, calcolare il nuovo blocco di parità e scriverlo
- un solo disco per la ridondanza (acceduto ad ogni write) diventa facilmente **bottleneck**
- si possono usare dischi hot spares
- RAID4 guasto \Rightarrow **due** dischi guasti insieme



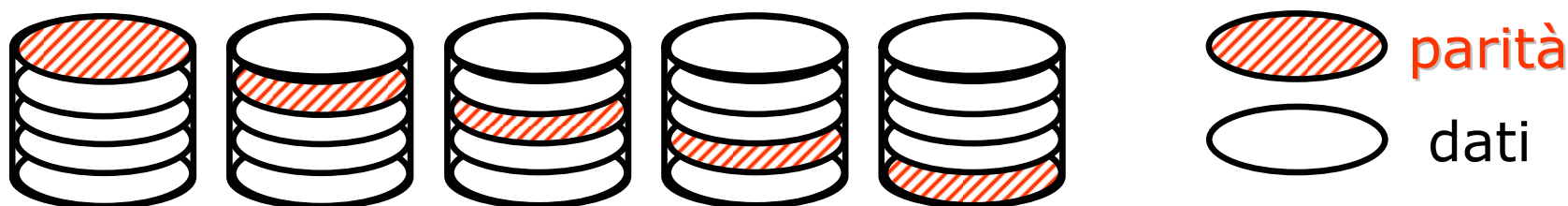
RAID level 4 - esempio

- 5 dischi fisici, uno usato per i dati ridondanti
- scrittura dei blocchi dati e di parità
- alta velocità (aggregata) di trasferimento in lettura - ridotta velocità di scrittura per la necessità di scrivere il blocco di parità

	Disco 1	Disco 2	Disco 3	Disco 4	Disco 5 ridondante
Stripe 1	<i>Block 1</i>	<i>Block 2</i>	<i>Block 3</i>	<i>Block 4</i>	Parity 1-4
Stripe 2	<i>Block 5</i>	<i>Block 6</i>	<i>Block 7</i>	<i>Block 8</i>	Parity 5-8
Stripe 3	<i>Block 9</i>	<i>Block 10</i>	<i>Block 11</i>	<i>Block 12</i>	Parity 9-12

RAID level 5: *block interleaved distributed parity*

- è la soluzione più adottata e versatile: massimi vantaggi (prestazioni e affidabilità) con minimi costi (limitata perdita di capacità, un solo disco)
- i blocchi di parità sono distribuiti uniformemente su tutti i dischi fisici (no bottleneck)
- le write sono più lente dei RAID0 e RAID1
- le read sono più veloci dei RAID1 (parallelismo)
- **load balancing** su tutti i dischi
- RAID5 guasto \Rightarrow **due** dischi guasti



RAID level 5: esempio

- 5 dischi fisici configurati come un disco logico RAID5
- la distribuzione dei blocchi dati e di parità è:

	Disco 1	Disco 2	Disco 3	Disco 4	Disco 5
Stripe 1	Block 1	Block 2	Block 3	Block 4	Parity 1-4
Stripe 2	Block 5	Block 6	Block 7	Parity 5-8	Block 8
Stripe 3	Block 9	Block 10	Parity 9-12	Block 11	Block 12

esecuzione di una write nel **Block 1** (1mo metodo)

- 1) read **Block 2,3** e 4
- 2) calcolo nuovo **Parity Block1-4** col nuovo **Block1**
- 3) write **Block 1** e **Parity Block 1-4**

RAID level 5: esempio (2)

	Disco 1	Disco 2	Disco 3	Disco 4	Disco 5
Stripe 1	Block 1	Block 2	Block 3	Block 4	Parity 1-4
Stripe 2	Block 5	Block 6	Block 7	Parity 5-8	Block 8
Stripe 3	Block 9	Block 10	Parity 9-12	Block 11	Block 12

esecuzione di una write nel *Block 1* (2do metodo)

- 1) read *Block 1, Parity 1-4*
- 2) calcolo del nuovo *Parity 1-4* da *old Block 1, new Block 1, old Parity 1-4*
- 3) write *new Block 1* e *Parity Block 1-4*

RAID level 5: esempio (3)

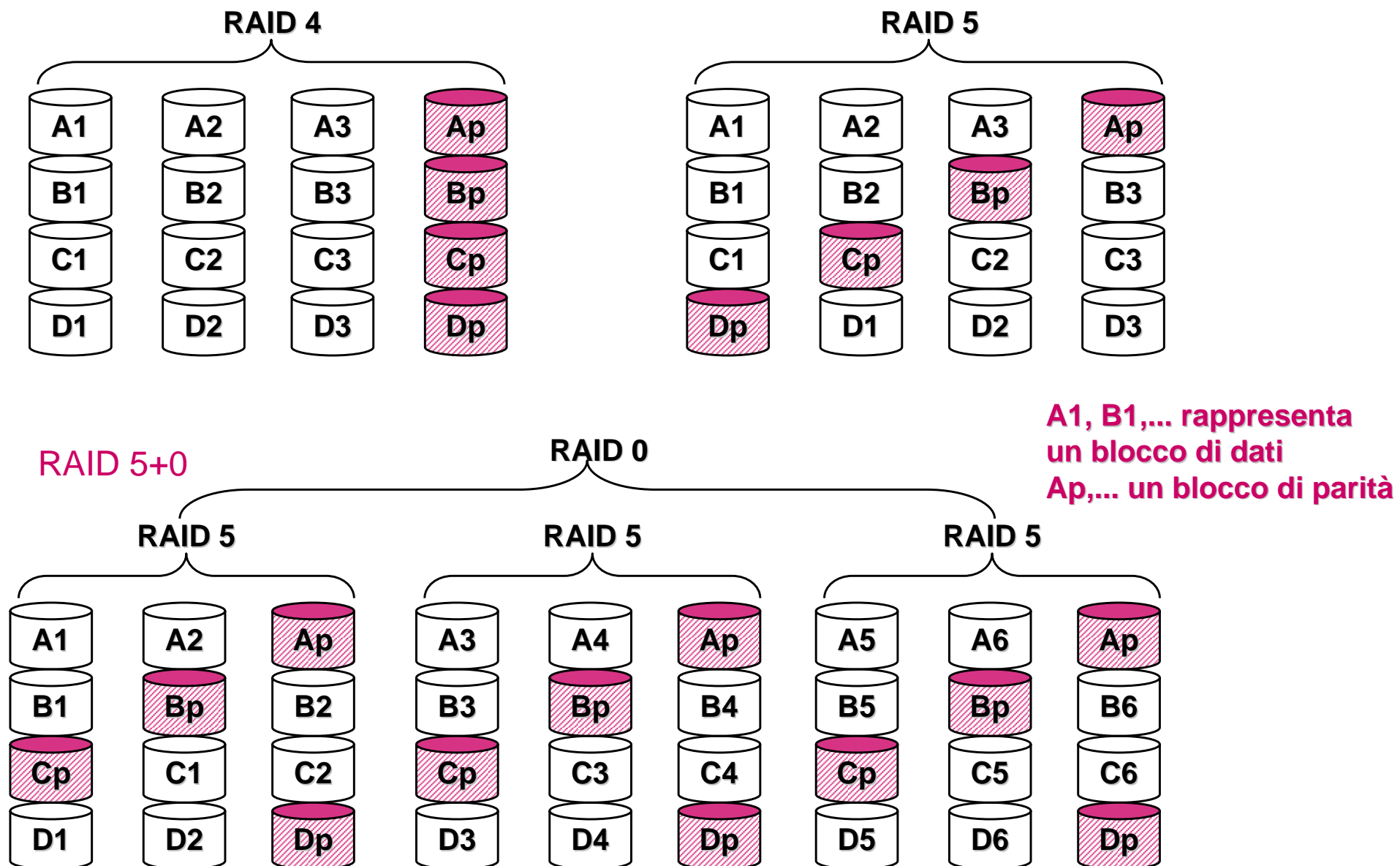
- **WRITE:** calcolo *new Parity 1-4* da *old Block1*, *old Parity 1-4*, *new Block1*

	Disco 1	Disco 2	Disco 3	Disco 4	Disco 5
Stripe 1	Block 1	Block 2	Block 3	Block 4	Parity 1-4
old	110	011	111	100	110
new	011	011	111	100	011

The diagram illustrates the XOR operation used in RAID 5 for a write. It shows a table with columns for five disks and rows for a stripe, old data, and new data. In the 'old' row, Block 1 is 110 and Parity 1-4 is 110. In the 'new' row, Block 1 is 011 and Parity 1-4 is 011. Dotted blue arrows indicate that the old Block 1 and old Parity 1-4 are XORed to produce the new Parity 1-4. A solid blue arrow indicates that the new Block 1 is XORed with the new Parity 1-4 to produce the new Block 2 (011).

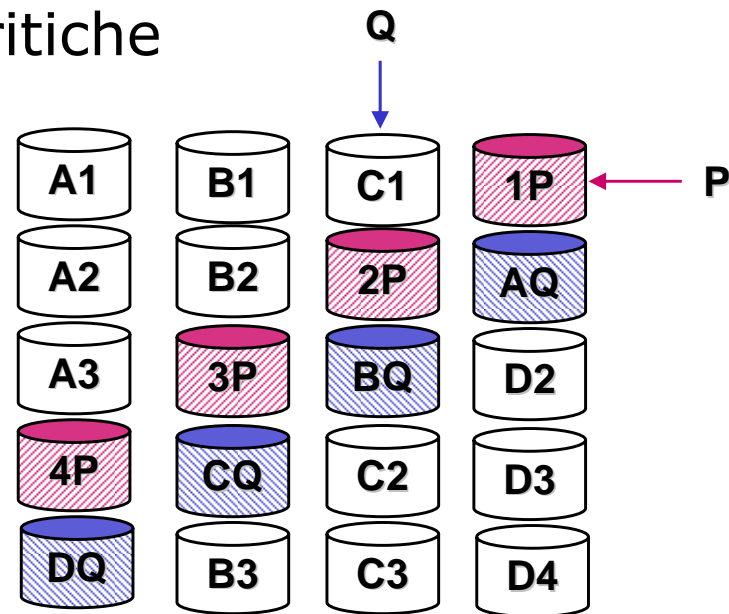
- **READ:** I blocchi di parità non sono acceduti nell'operazione di lettura. Tuttavia vengono letti se un settore dà luogo a un errore CRC (cyclic redundancy check), in questo caso il settore errato viene ricostruito dai rimanenti blocchi della stripe e dal blocco di parità

schema RAID 4, 5, 5+0



RAID level 6

- estensione di RAID 5 ma con maggiore *fault tolerance*
- due sistemi di parità (P+Q) distribuiti e indipendenti
- tollera il guasto simultaneo di 2 dischi
- elevato overhead per calcolo della parità
- richiede $N + 2$ dischi
- prestazioni povere in scrittura
- adatto per applicazioni molto critiche
- difficile da realizzare



criteri di scelta del livello di RAID

■ Velocità

- I/O di scrittura
- I/O di lettura
- tempi di recovery

■ Parallelismo

■ Affidabilità

- Fault-tolerance
- correzione errori

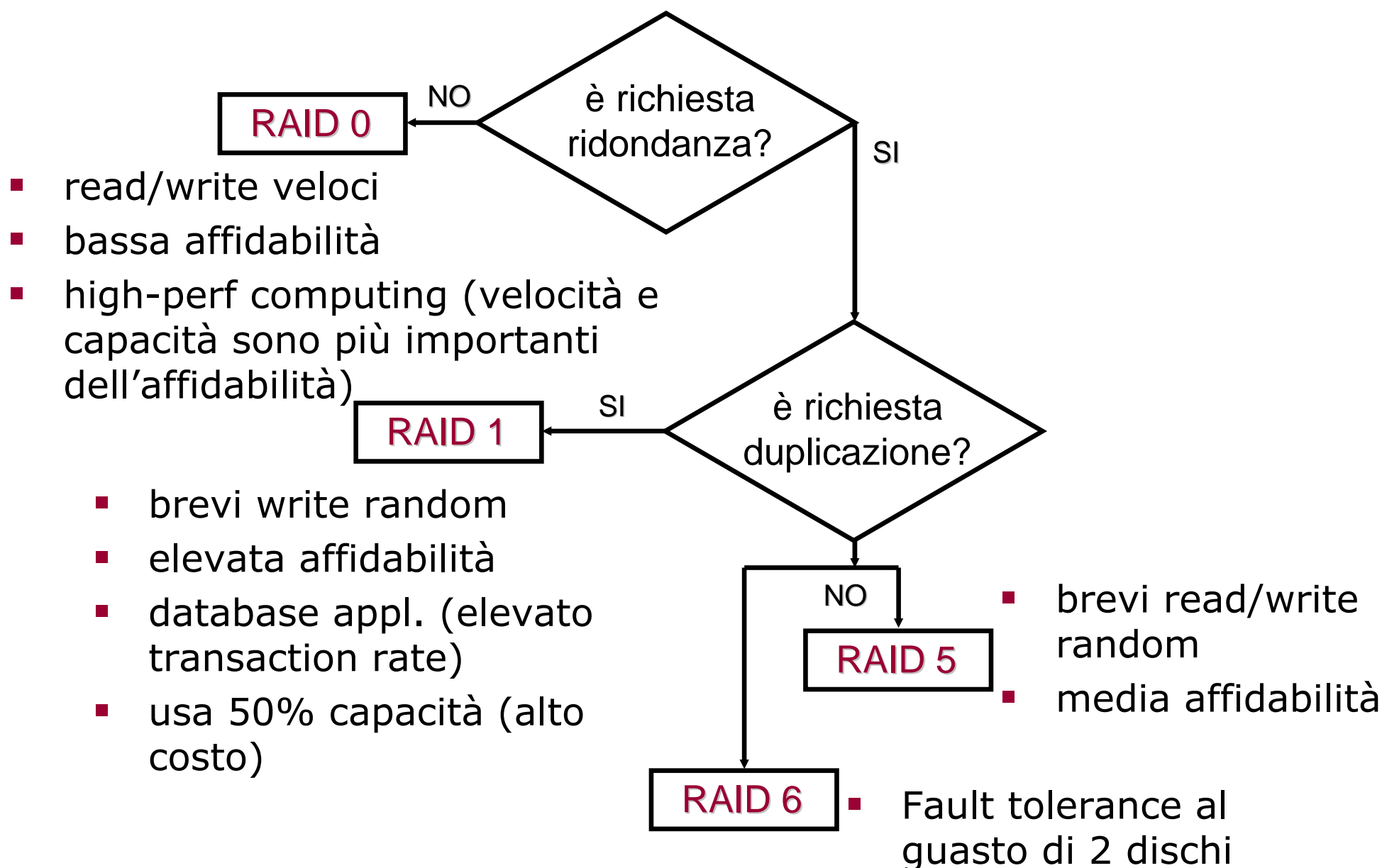
■ Ridondanza

■ Duplicazione

■ Costi

- sfruttamento della capacità fisica
- tipi di soluzione
- caratteristiche controller

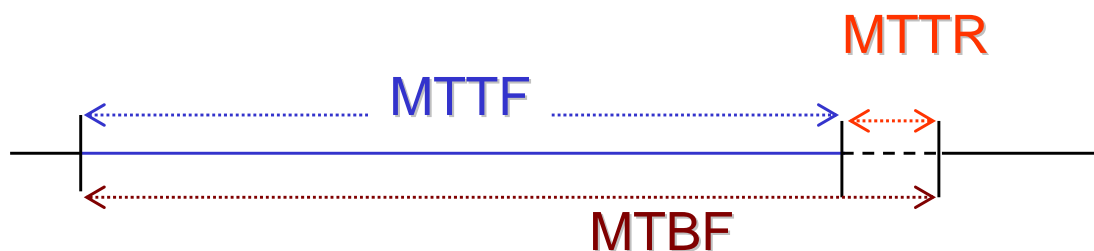
criteri di scelta del livello di RAID



Affidabilità

(MTTF) Mean Time To Failure

- **MTBF** (Mean-Time-Between-Failures)
 - tempo medio che intercorre tra due guasti
- **MTTF** (Mean-Time-To-Failures)
 - tempo medio perché si verifichi un guasto
 - l'unità di misura tipica sono i FIT (failure in time)
 - numero di guasti ogni miliardo di ore (10^{-9} ore $^{-1}$)
 - si ipotizza che un prodotto attraversi ciclicamente guasti e riparazioni
 - si suppone che questi cicli avvengano in modo casuale e mediamente stazionario
 - dopo una riparazione il prodotto è come nuovo



MTTR (Mean Time To Repair)

- **MTTR** (Mean-Time-To-Repair)
 - tempo medio per riparare un prodotto guasto
 - rappresenta il tempo medio durante il quale il prodotto non funziona
 - comprende il tempo necessario per
 - scoprire che c'è un guasto
 - individuare il guasto
 - rimuovere il componente difettoso
 - effettuare la riparazione
 - ripristinare il componente
 - effettuare le operazioni software per ripristinare il sistema
- bassi MTTR sono molto costosi

possibili cause per i guasti hardware

- **Design failures:** guasti causati da errori di progetto o costruzione. Nei sistemi ben fatti questa categoria di guasti dovrebbe contribuire in modo trascurabile al totale dei guasti
- **Infant Mortality:** guasti che si presentano nei sistemi nuovi. Normalmente questa categoria di guasti non dovrebbe presentarsi nei sistemi in produzione ma emergere durante le fasi di test
- **Random Failures:** guasti casuali si presentano durante l'intera vita di un sistema. Questa è la categoria di guasti che viene tipicamente considerata negli studi di affidabilità
- **Wear Out:** quando un sistema ha raggiunto la fine della sua vita utile, la degradazione di alcune componenti causa il guasto del sistema. La manutenzione preventiva può ritardare l'insorgere di questi guasti

possibili cause per i guasti software

- I *guasti software* sono caratterizzati dalla *densità* di difetti nel sistema. La densità dei difetti dipende dai seguenti fattori
 - Il processo utilizzato per sviluppare le applicazioni (l'utilizzo o meno di *unit testing*, ...)
 - Complessità del software
 - Dimensione del software
 - Esperienza del team di sviluppo
 - Percentuale di codice riutilizzato da altre applicazioni *stabili*
 - Rigore della metodologia di testing funzionale prima del rilascio in produzione
 - La densità di difetti è misurata tipicamente come numero di difetti per migliaia di linee di codice (defects/KLOC)

come stimare MTTR (software)

- la MTTR per un modulo software può essere definita come il tempo necessario per riavviare l'applicativo dopo che si è individuato il guasto
 - molte failure software vengono eliminate riavviando l'applicazione o l'intero sistema
 - questo non corregge il motivo che ha generato la failure

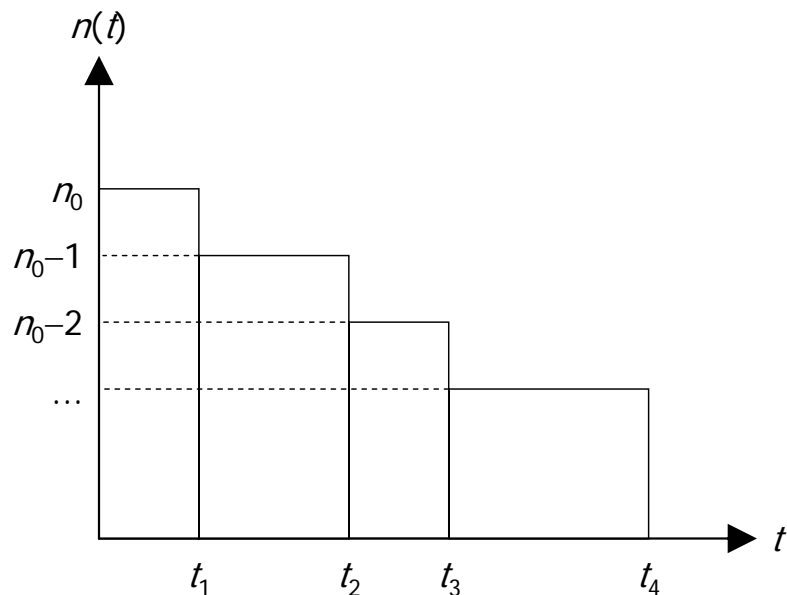
availability (disponibilità)

- L'availability di un sistema è la percentuale di tempo in cui il sistema funziona correttamente
 - probabilità (media stazionaria) che in un qualsiasi istante il sistema sia *funzionante*

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

- una notazione tipica è quella dei *nove*
 - un'affidabilità a 3-nove corrisponde al 99.9%
 - un'affidabilità a 5-nove corrisponde al 99.999%

reliability (affidabilità) e failure rate



- Consideriamo n_0 elementi indipendenti e statisticamente identici che vengano messi in esercizio al tempo $t=0$ alle stesse condizioni
 - $n(0) = n_0$
- Al tempo t un sottoinsieme $n(t)$ di questi elementi *non si sono ancora guastati*
- t_1, t_2, \dots, t_{n_0} sono i tempi al guasto osservati sugli n_0 elementi
- I tempi al guasto sono realizzazioni indipendenti della variabile casuale τ (tempo al guasto dell'elemento)
- $E[\tau] = (t_1 + t_2 + \dots + t_{n_0}) / n$
- è la media empirica di τ
- Per $n \rightarrow \infty$ converge al valore $E[\tau] \rightarrow \text{MTTF}$
- La funzione $n(t) / n_0$ è la funzione empirica di affidabilità
- Per $n \rightarrow \infty$ converge al valore
 - $n(t) / n_0 \rightarrow R(t)$

reliability e failure rate (2)

- X : istante di guasto di un componente
- $F(t)$: sua distribuzione cumulativa
- $R(t)$: probabilità che il componente (perfettamente funzionante al tempo 0) sia ancora funzionante al tempo t
- $R(t) = P(X > t) = 1 - F(t)$
- $\lambda(t)$: **failure rate** - probabilità che il componente si guasti all'istante t

$$\lambda(t) = \lim_{\Delta t \rightarrow 0} \frac{P(X \in (t, t + \Delta t) | X > t)}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{P(X \in (t, t + \Delta t) \cap X > t)}{\Delta t} \cdot \frac{1}{P(X > t)}$$

$$= \lim_{\Delta t \rightarrow 0} \frac{P(X \in (t, t + \Delta t))}{\Delta t \cdot (1 - F(t))} = \frac{f(t)}{R(t)} = -\frac{1}{R(t)} \frac{dR(t)}{dt}$$

se $F(t)$ è esponenziale:
 λ = parametro (costante)
della distribuzione

reliability e failure rate (3)

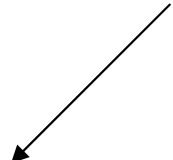
- $f(t)dt$ è la probabilità di guasto in $(t, t+dt)$
- la media della distribuzione $F(t)$ è la:
 - **MTTF (Mean Time To Failure)**
- se supponiamo $F(t)$ esponenziale:

$$F(t) = P(X \leq t) = 1 - e^{-\frac{t}{MTTF}}$$

$$\cong \frac{t}{MTTF} \quad \left(per \frac{t}{MTTF} \ll 1 \right)$$

$$\boxed{\frac{1}{MTTF}}$$

Probabilità di guasto
nell'unità di tempo



componenti in parallelo

- affidabilità di un sistema calcolata da quella dei componenti, considerati *indipendenti* (il guasto di un elemento non rende più o meno facile quello degli altri)

$$S = \max\{x_1, \dots, x_n\}$$

$$F_S(t) = \prod_{i=1}^n F_i(t)$$

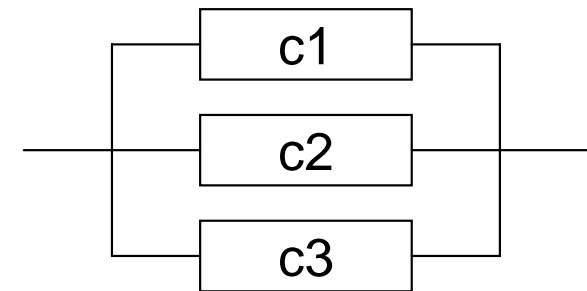
$$F_S(t) = P(S \leq t)$$

$$= P(x_1 \leq t \cap x_2 \leq t \dots)$$

$$= P(x_1 \leq t) \cdot P(x_2 \leq t) \cdot \dots$$

$$= F_1(t) \cdot F_2(t) \cdot \dots$$

$$R_S(t) = 1 - \prod (1 - R_i(t))$$

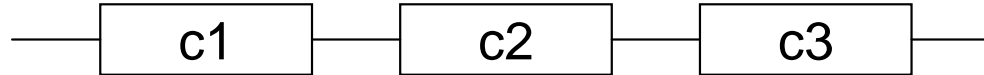


← indipendenza

← definizione

- distribuzione del *massimo* fra n:
 - il sistema si guasta quando si guasta l'ultimo componente (n componenti in parallelo)

componenti in serie



$$S = \min \{x_1, \dots, x_n\} \quad 1 - \prod_{i=1}^n R_i(t)$$

$$R_S(t) = \prod_{i=1}^n R_i(t)$$

$$F_S(t) = P(S \leq t)$$

$$= P(x_1 \leq t \cup x_2 \leq t \dots)$$

$$= 1 - P(x_1 > t \cap x_2 > t \dots) \quad \longleftarrow \text{dualità}$$

$$= 1 - (1 - P(x_1 \leq t))(1 - P(x_2 \leq t)) \dots \quad \longleftarrow \text{indipendenza}$$

$$= 1 - \prod_{i=1}^n (1 - F_i(t)) \quad \longleftarrow \text{definizione}$$

- distribuzione del *minimo* fra n:
 - il sistema si guasta quando si guasta il primo componente (n componenti in serie)

affidabilità di un sistema - riassunto in formule

condizioni di guasto

$$\prod_{i=1}^n F_i(t)$$

- *tutti* i componenti

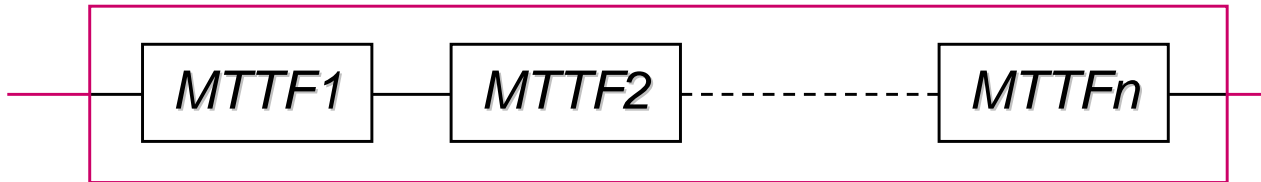
$$1 - \prod_{i=1}^n (1 - F_i(t))$$

- *un* componente

$$\sum_{i=k}^n \binom{n}{i} F(t)^i (1 - F(t))^{n-i}$$

- *k su n* componenti (di identiche caratteristiche)

affidabilità di un sistema - MTTF (n componenti in serie)



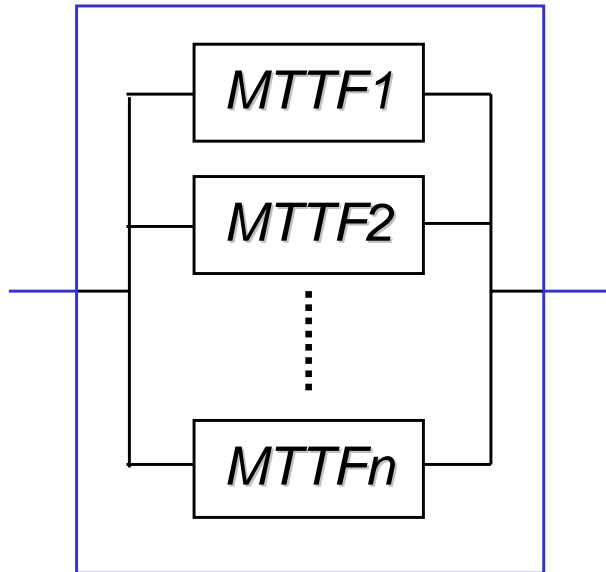
$$F_{SERIE} = 1 - \prod_{i=1}^n (1 - F_i(t)) = 1 - \left(e^{-\left(\frac{1}{MTTF1} + \frac{1}{MTTF2} + \dots \right) t} \right)$$

$$MTTF_{SERIE} = \frac{1}{\frac{1}{MTTF1} + \frac{1}{MTTF2} + \dots + \frac{1}{MTTFn}}$$

$$MTTF_{SERIE} = \frac{MTTF}{n}$$

se i componenti sono identici

affidabilità di un sistema - MTTF (n componenti in parallelo)



consideriamo solo il caso di n componenti identici:

il primo si guasta mediamente dopo un tempo $t_1 = MTTF/n$

il secondo dopo un tempo (a partire dall'inizio)

$t_2 = t_1 + MTTF/(n-1)$

infatti

in $(0, t_1)$ abbiamo n componenti in funzione,

in (t_1, t_2) $n - 1$ e così via

in (t_{n-1}, t_n) uno solo

$$MTTF_{PARALLELO} = MTTF \left(\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{2} + 1 \right)$$

metriche utilizzate

- L'affidabilità di un RAID livello 0 con n dischi è bassa, infatti:

$$\text{MTTDL}_n = \text{MTTF}_1 / n$$

- **MTTF₁**: Mean Time To Failure, tempo medio che intercorre prima che un disco si guasti
- **MTTDL_n**: Mean Time To Data Loss, tempo medio prima che si guastino più dischi dell'array causando perdita **irrimediabile** di dati

calcolo MTTF di n dischi

- assumendo che i tempi tra due guasti successivi siano distribuiti esponenzialmente:
 - $MTTF(\text{array}) = MTTF(\text{disco}) / \text{numero dischi}$

$$P(\min(X) \leq t) = 1 - (1 - F(t))^n = 1 - \left(e^{-\frac{t}{MTTF}} \right)^n \cong \frac{n}{MTTF} \cdot t$$

$$MTTF(n \text{ dischi}) = MTTF(\text{disco}) / n$$

guasto di un array

- G: numero di dischi per gruppo di parità
- un gruppo si guasta effettivamente se si guasta un secondo disco durante l'intervallo di tempo necessario alla riparazione del primo

$$P(2do\ guasto) = 1 - \left(e^{-\frac{MTTR}{MTTF}} \right)^{G-1} \cong \frac{MTTR (G-1)}{MTTF}$$

calcolo MTDDL

- Perciò:

$$\begin{aligned} MTTF_{Gruppo} &= \frac{MTTF_{disco}}{G} \cdot \frac{MTTF_{disco}}{MTTR \cdot (G-1)} \\ &= \frac{(MTTF_{disco})^2}{G(G-1) \cdot MTTR} \end{aligned}$$

- se N è il numero totale di dischi dell'array

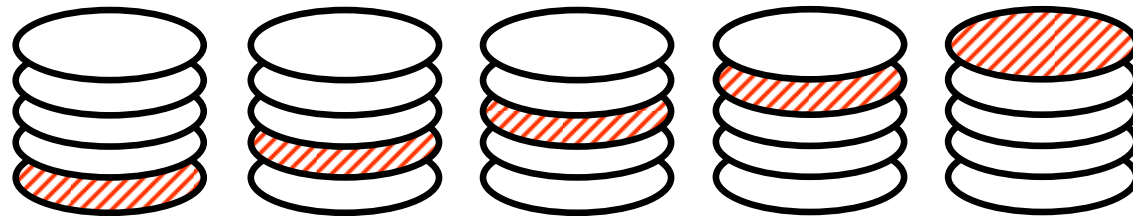
$$MTTF_{array} = \frac{MTTF_{Gruppo}}{\text{Numero gruppi}} = \frac{(MTTF_{disco})^2}{N(G-1) \cdot MTTR}$$

- in modo analogo se ci fossero due dischi di parità per gruppo (cioè i dati si perdono se si guasta un terzo disco)

$$MTTF_{array} = \frac{(MTTF_{disco})^3}{N(G-1)(G-2) \cdot MTTR^2}$$

case study: calcolo MTDDL di un RAID5 (1)

- **MTTF (*disco*)**: Mean Time To Failure di un disco
- **MTTR (*disco*)**: Mean Time To Repair di un disco
- **MTDDL (*RAID5*)**: Mean Time To Failure di un RAID5



- $N=25$ dischi totali \rightarrow 5 gruppi da $G=5$ dischi
- ogni gruppo ha un disco ridondante
- $MTTF(\text{disk})=1000$ giorni, $MTTR(\text{disk})=10$ giorni

calcolo MTDDL di un RAID5 (2)

$$prob(un\ disco\ guasto) = \frac{1}{MTTF(disco)}$$

- prob. che uno dei 5 dischi di un gruppo si guasti:

$$\begin{aligned} prob(un\ guasto\ nel\ gruppo) &= G \times prob(un\ disco\ guasto) = \\ &= \frac{G}{MTTF(disco)} = \frac{5}{1000} = 0.005 \end{aligned}$$

- prob. che uno dei rimanenti 4 dischi di un gruppo si guasti durante i 10 giorni di riparazione del primo disco guasto:

$$\begin{aligned} prob(un\ gruppo\ guasto) &= \frac{G}{MTTF(disco)} \left[\frac{G-1}{MTTF(disco)} MTTR \right] = \\ &= \frac{5}{1000} \frac{4}{1000} 10 = 0.0002 \end{aligned}$$

calcolo MTDDL di un RAID5 (3)

- probabilità che uno tra i 5 gruppi sia guasto

$$\begin{aligned} \text{prob}(\text{RAID5 guasto}) &= \text{num. gruppi} \times \text{prob}(\text{un gruppo guasto}) = \\ &= \frac{\text{num. gruppi} \times G \times (G - 1) \times \text{MTTR}(\text{disco})}{\text{MTTF}(\text{disco})^2} = \\ &= \frac{N \times (G - 1) \times \text{MTTR}(\text{disco})}{\text{MTTF}(\text{disco})^2} = 5 \times 0.0002 = 0.001 \end{aligned}$$

$$\text{MTTF}(\text{RAID5}) = \frac{1}{\text{prob}(\text{RAID5 guasto})} = \frac{1}{0.001} = 1000 \text{ giorni}$$