componenti storage

M. Arrigoni Neri & *P. Borghese*

indice

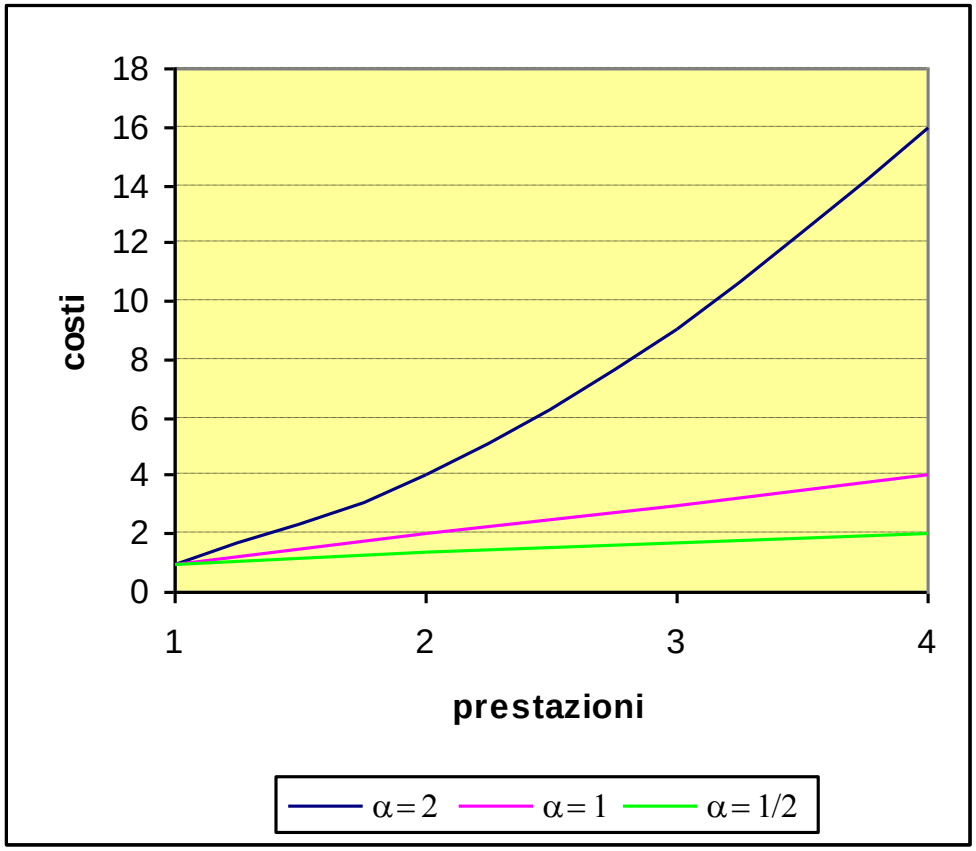
- leggi empiriche di evoluzione
- gerarchie di memoria
- architettura dischi
- prestazioni delle operazioni di I/O
- metodi per migliorare le prestazioni di I/O

leggi di evoluzione

scalabilità e costi (legge di Grosch)

- relazione empirica fra costo *c* e capacità operativa (prestazione) *w*

$$c \approx w^{\alpha}$$



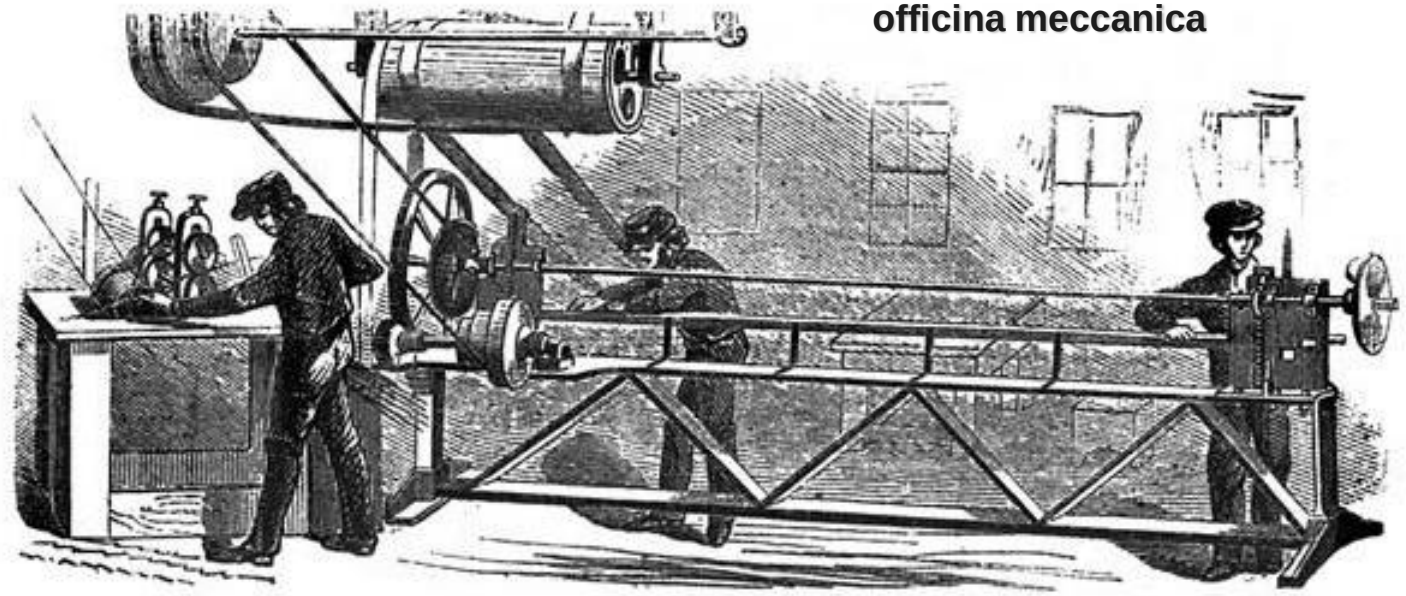
$$c^{\frac{1}{\alpha}} = w$$

$$\frac{w}{c} = c^{\left(\frac{1}{\alpha}-1\right)}$$

scalabilità e costi (legge di Grosch) (cont.)

- se $\alpha < 1$ si verifica una economia di scala, perciò conviene concentrare la capacità in uno o pochi componenti.

“main frame” in una officina meccanica

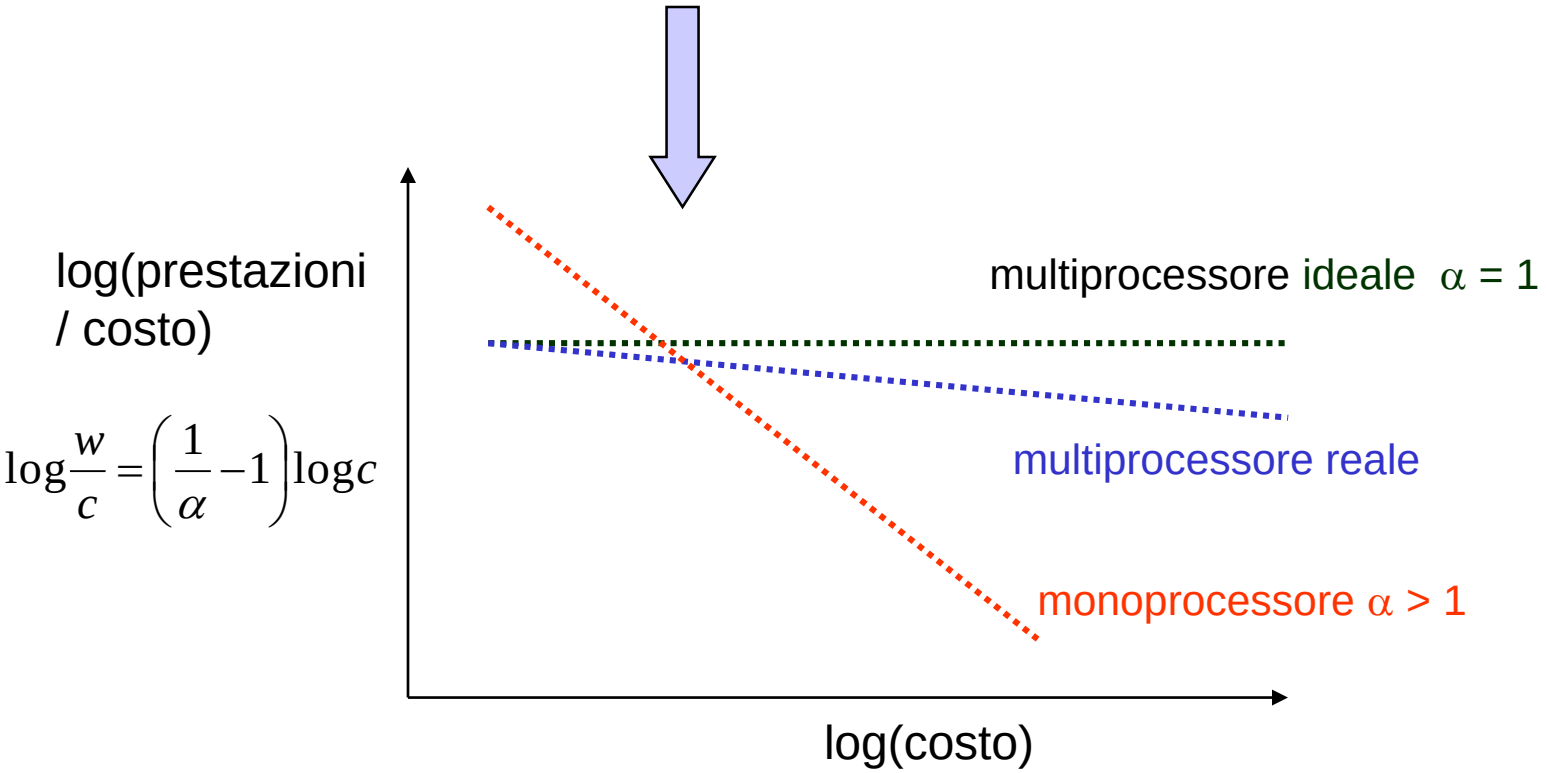


fonte: Scientific American, 1859

illustrazione storica di tecnologie ora superate in cui si verificava il caso $\alpha < 1$

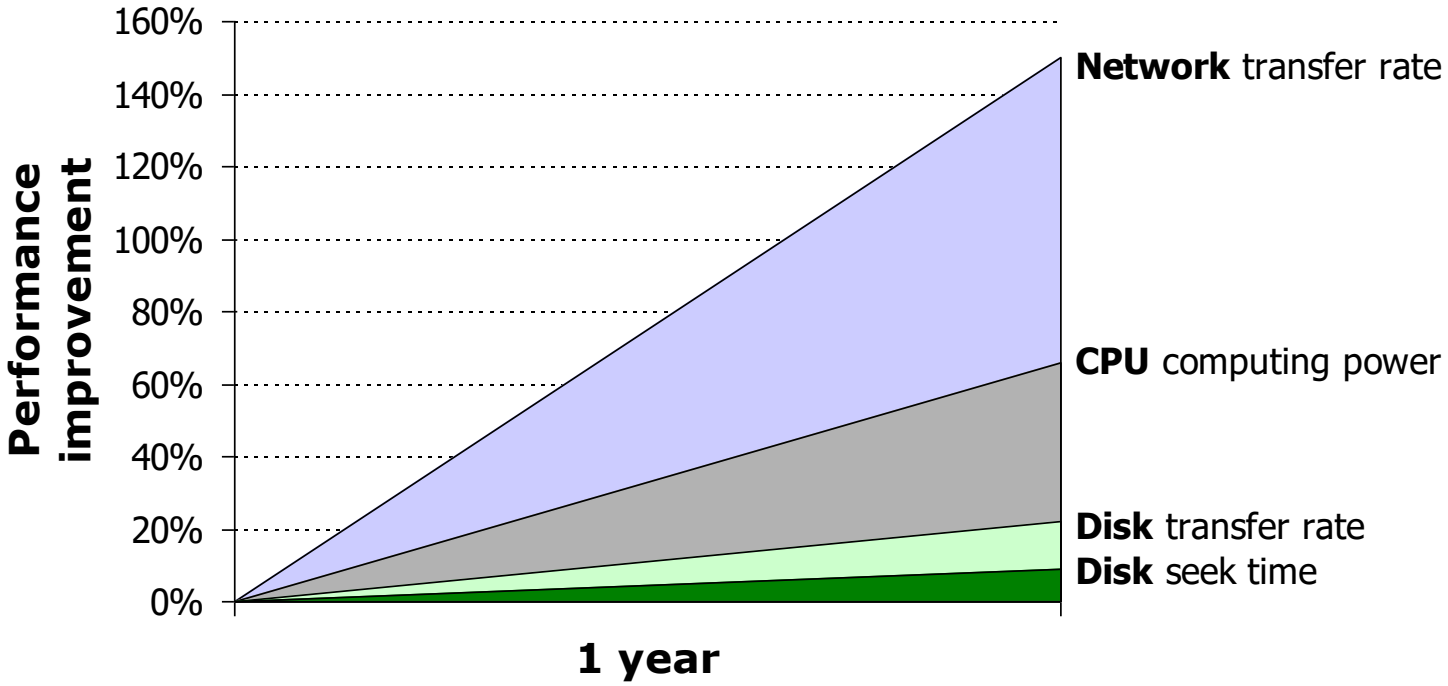
scalabilità e costi (legge di Grosch) (cont.)

- in genere per tecnologie mature $\alpha > 1$
- attualmente per i componenti informatici $\alpha > 1$, perciò è conveniente distribuire la capacità.



evoluzione delle capacità (legge di Moore)

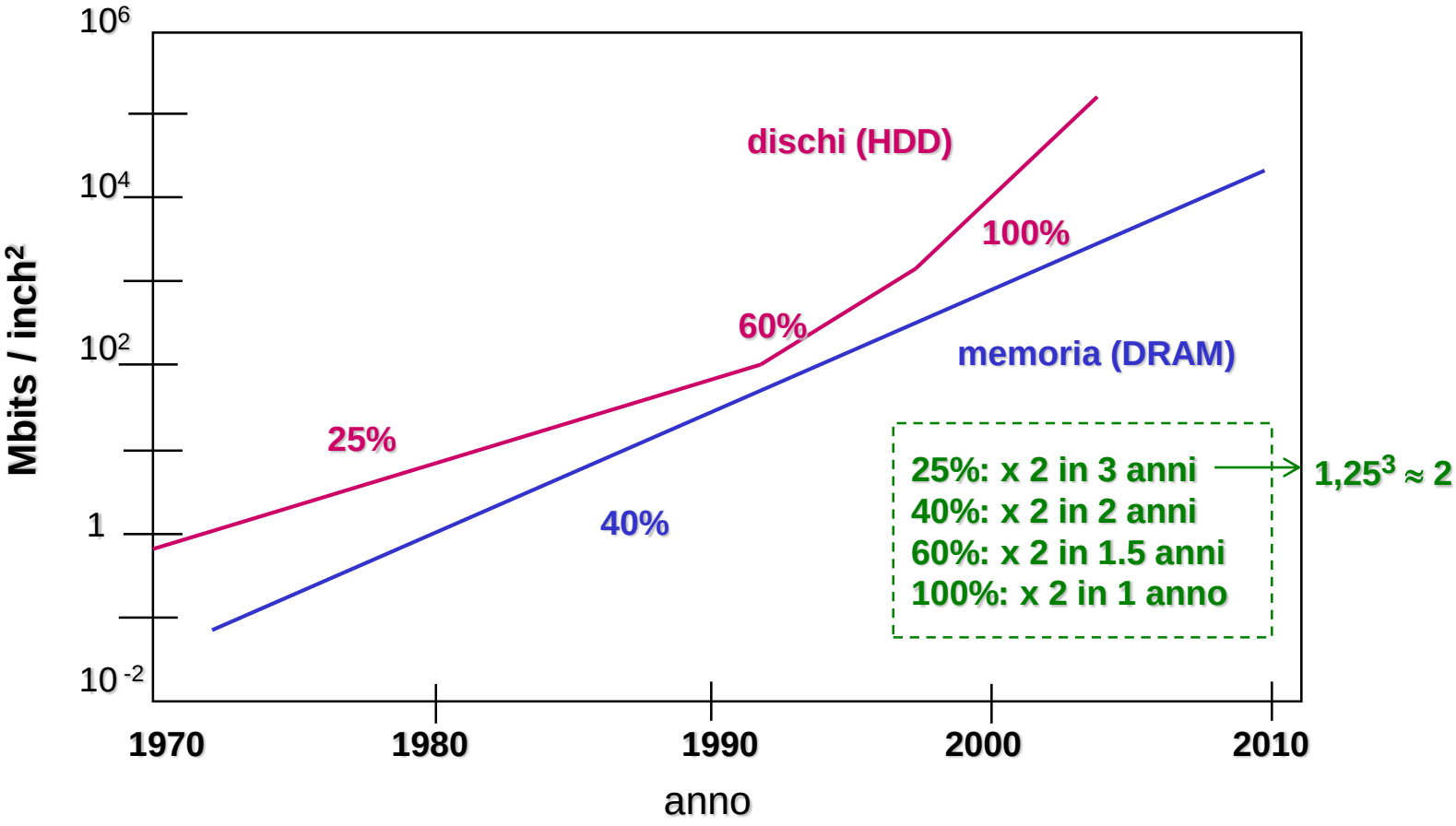
- legge empirica che all'origine riguardava l'andamento della densità dei transistori per chip (che raddoppia ogni 18 mesi)
- concerne l'incremento della *capacità operativa* (che si moltiplica approssimativamente per 100 ogni 10 anni)



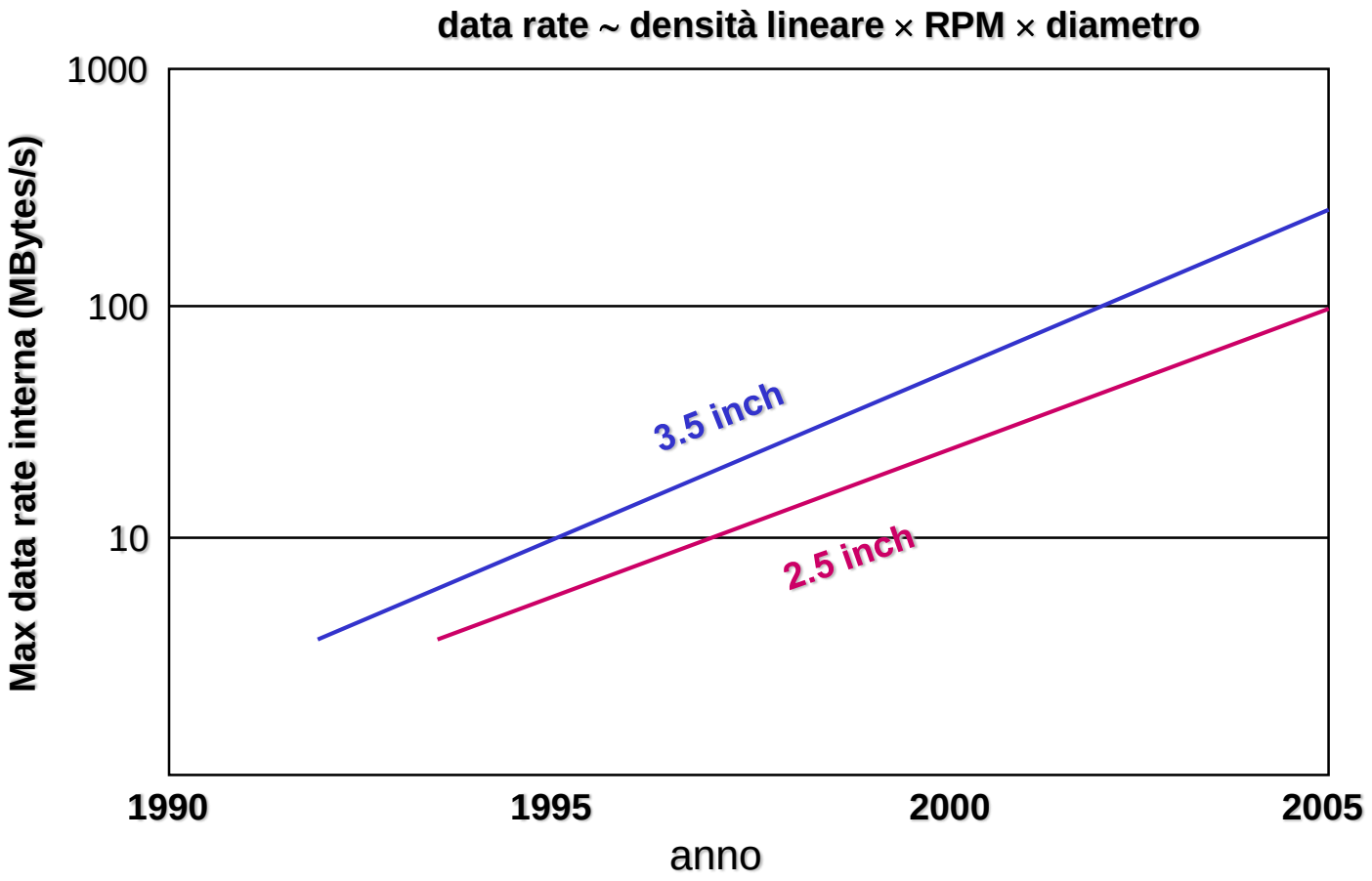
evoluzione delle capacità (legge di Moore) (cont.)

- CGR (Compound Growth Rate – tasso composto di crescita)
  - $g(t) = \alpha^t$ (t: anni)
  - α: coefficiente di incremento unitario (per anno)
  - $\log g(t) = t \log \alpha$
    - t2: tempo di raddoppio (g(t2)=2)
    - $\log 2 = t2 \log \alpha$
    - t2 = 1.5 anni (18 mesi)
  - $\log 2 = 1.5 \log \alpha$
  - $\alpha = 1.587$
  - $g(10) = (1.587)^{10} = 101.33$
- Il 1mo disco è del 1956, l'architettura è cambiata poco negli anni, i progressi si sono realizzati soprattutto nella geometria (miniaturizzazione) e nelle tecnologie

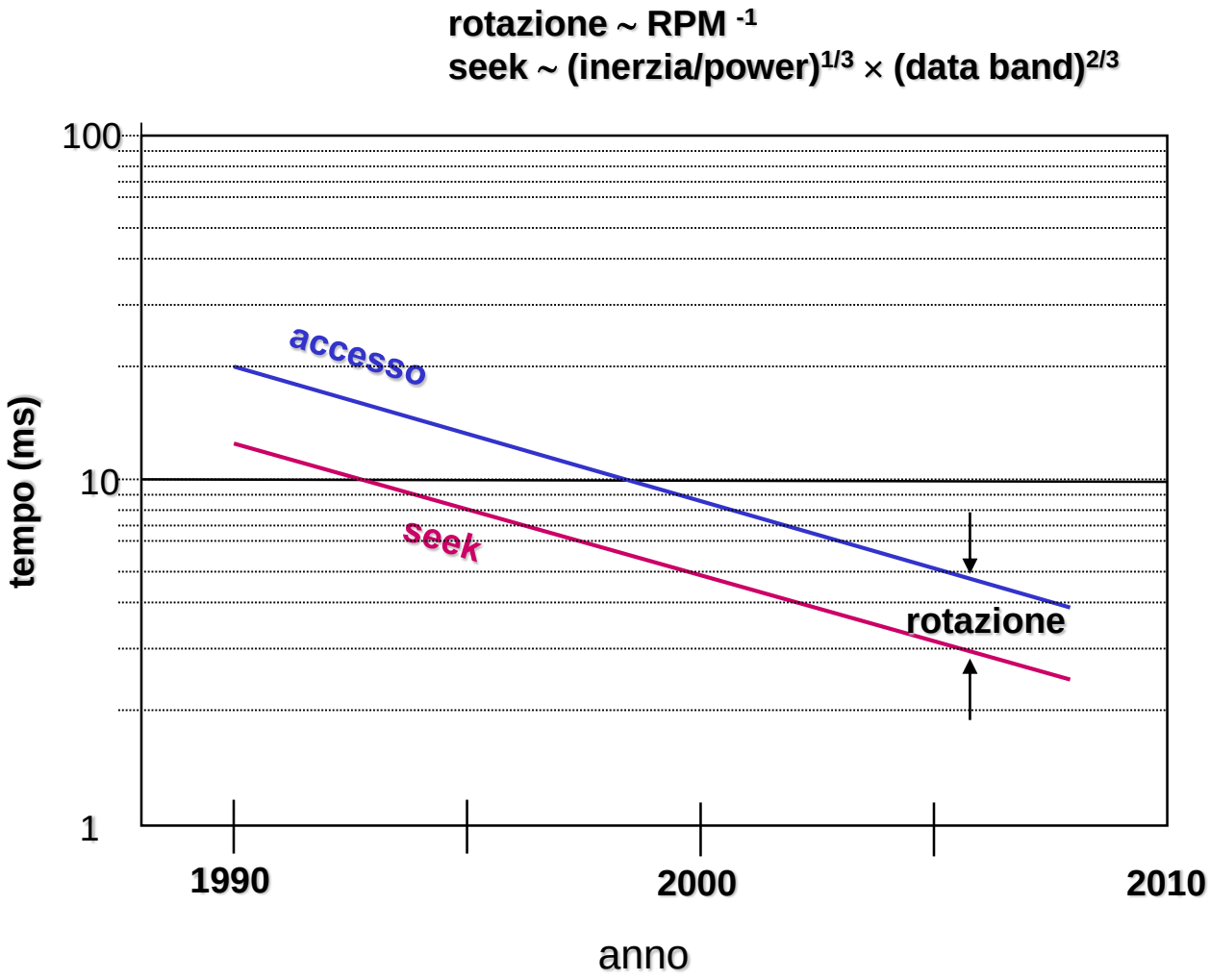
esempio: evoluzione dischi (densità superficiale)



esempio: evoluzione dischi (data rate)

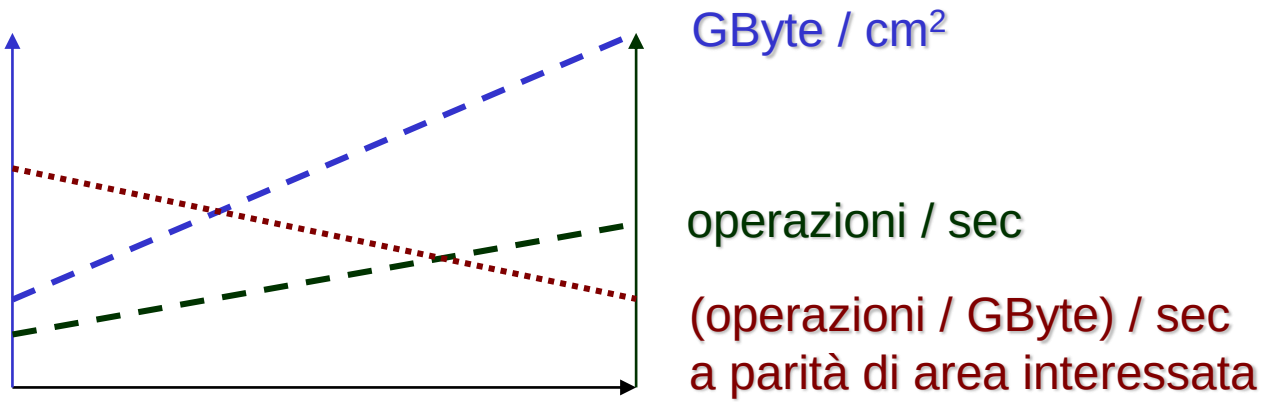


esempio: evoluzione dischi (tempi di accesso)



osservazione

- la crescita di alcune grandezze di prestazione con *tassi diversi* fra loro può dare luogo a problemi di "non allineamento":
  - 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 al 2005 di 5×10^7 volte,
- con metodi olografici 1 Tbit può essere contenuto in un volume di 1 cm^3 (*fonte: Scientific American aug. 05*)

gerarchie di memoria

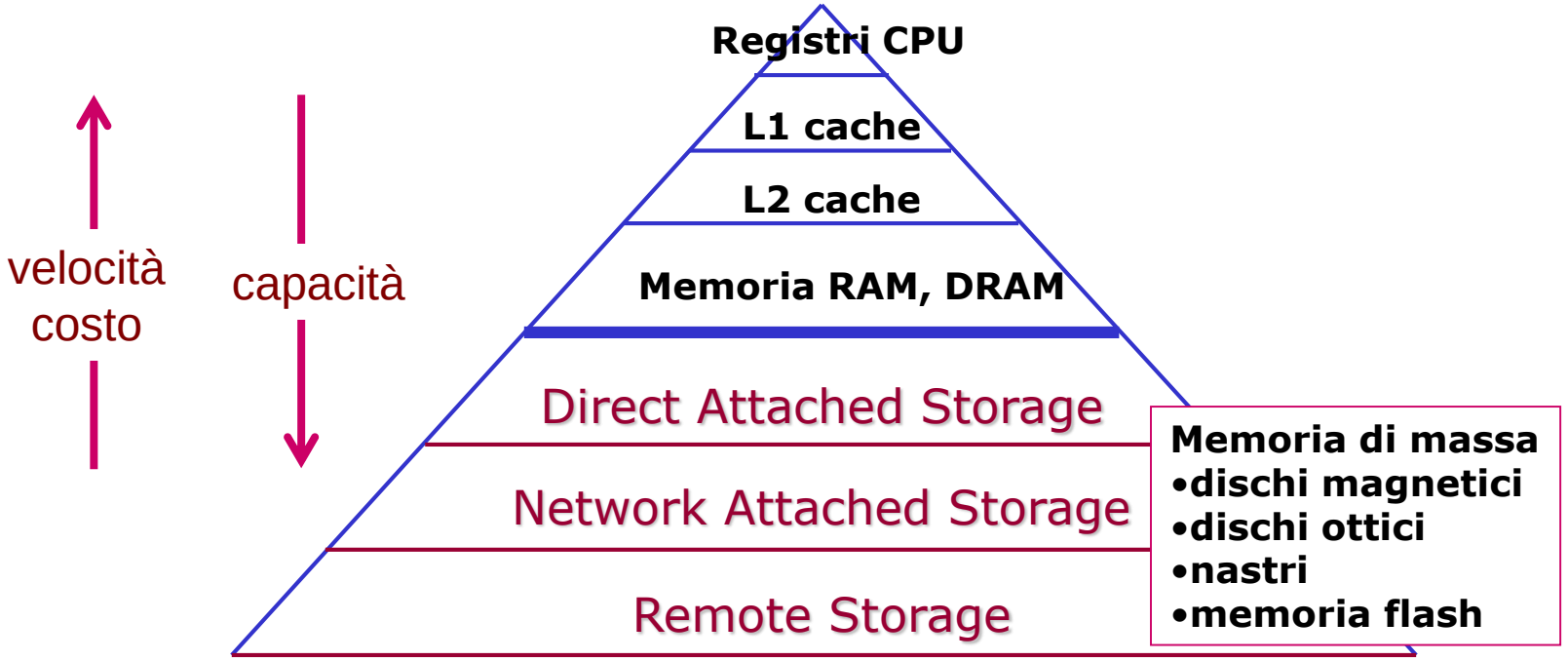
(metodo per comporre velocità diverse dei dispositivi)

L'architettura di un computer è composta da una serie di "cache" sovrapposte che sfruttano la località spaziale e temporale dei dati e degli algoritmi.

La storia dell'informatica può essere interpretata anche come storia della virtualizzazione.

livelli gerarchici di memoria

- le prestazioni dei dischi sono critiche per le prestazioni globali di un sistema e crescono nel tempo con un tasso inferiore a quelle dei processori
- le gerarchie di memoria sono una forma di *virtualizzazione* dei dati



esercizio 1: calcolo del tempo medio di accesso

Esempio di calcolo del tempo medio di accesso al dato

	layer	*tempo* ***t***	*miss rate* ***m***	*prob.* ***p***	*p x t*
1	*Reg*	1,00E+00	1,00E-01	1,00E+00	1,00E+00
2	*L1*	1,00E+00	5,00E-02	1,00E-01	1,00E-01
3	*L2*	8,00E+00	2,00E-02	5,00E-03	4,00E-02
4	*Main mem.*	1,00E+02	1,00E-01	1,00E-04	1,00E-02
5	*Local disk*	1,00E+07	2,00E-02	1,00E-05	1,00E+02
6	*Net server*	5,00E+07	2,00E-02	2,00E-07	1,00E+01
7	*Remote server*	4,00E+08	0,00E+00	4,00E-09	1,60E+00
		tempo di accesso al componente in unità ipotetiche		***tot***	**112,75**

$p(i) = p(i-1) \times m(i-1)$ *probabilità di arrivare nella ricerca del dato al livello (i)*

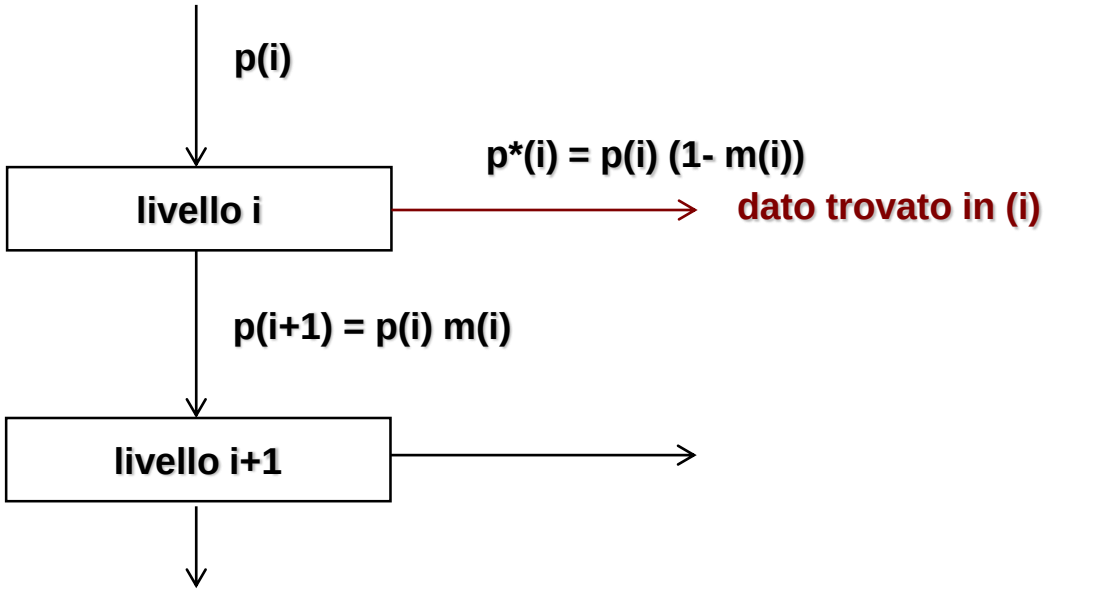
$p(i) \times t(i)$ *contributo al tempo di accesso del livello (i)*

tempo totale $= \Sigma\ p(i) \times t(i)$

osservazione sulle probabilità e loro calcolo

- gli *eventi* E(i) a cui competono le probabilità p(i), cioè le visite ai diversi livelli della gerarchia di memoria, *non sono esclusivi* infatti:
  - $E(k) \subseteq E(j)$; $p(k) \leq p(j)$ per $k > j$ - la visita al livello (k) richiede di essere prima passati da (j)
  - $\Sigma\ p(i) > 1$
- passiamo a costruire degli *eventi incompatibili* E*(i):
  - E*(i): il dato cercato si trova al livello (i)
  - $p^*(i) = (1 - m(i)) \times p(i)$ - probabilità che la ricerca si arresti al livello (i)
  - $\Sigma\ p^*(i) = 1$
  - t*(i): tempo per ottenere il dato residente al livello (i) = somma dei tempi di accesso ai livelli da 1 a (i)

osservazione sulle probabilità e loro calcolo (cont.)



- la tabella dell'esercizio 1 si trasforma in quella della pagina seguente (esercizio 2)
  - ovviamente i due metodi devono portare allo stesso risultato finale (tempo medio)

esercizio 2

Esempio di calcolo del tempo medio di accesso al dato (variazione sul metodo)

	layer	*miss rate* ***m***	***p****	***t****	*p* x t**
1	*Reg*	1,00E-01	9,00E-01	1,00E+00	9,00E-01
2	*L1*	5,00E-02	9,50E-02	2,00E+00	1,90E-01
3	*L2*	2,00E-02	4,90E-03	1,00E+01	4,90E-02
4	*Main mem.*	1,00E-01	9,00E-05	1,10E+02	9,90E-03
5	*Local disk*	2,00E-02	9,80E-06	1,00E+07	9,80E+01
6	*Net server*	2,00E-02	1,96E-07	6,00E+07	1,18E+01
7	*Remote server*	0,00E+00	4,00E-09	4,60E+08	1,84E+00
			Σ *p* = 1*	***tot***	**112,75**

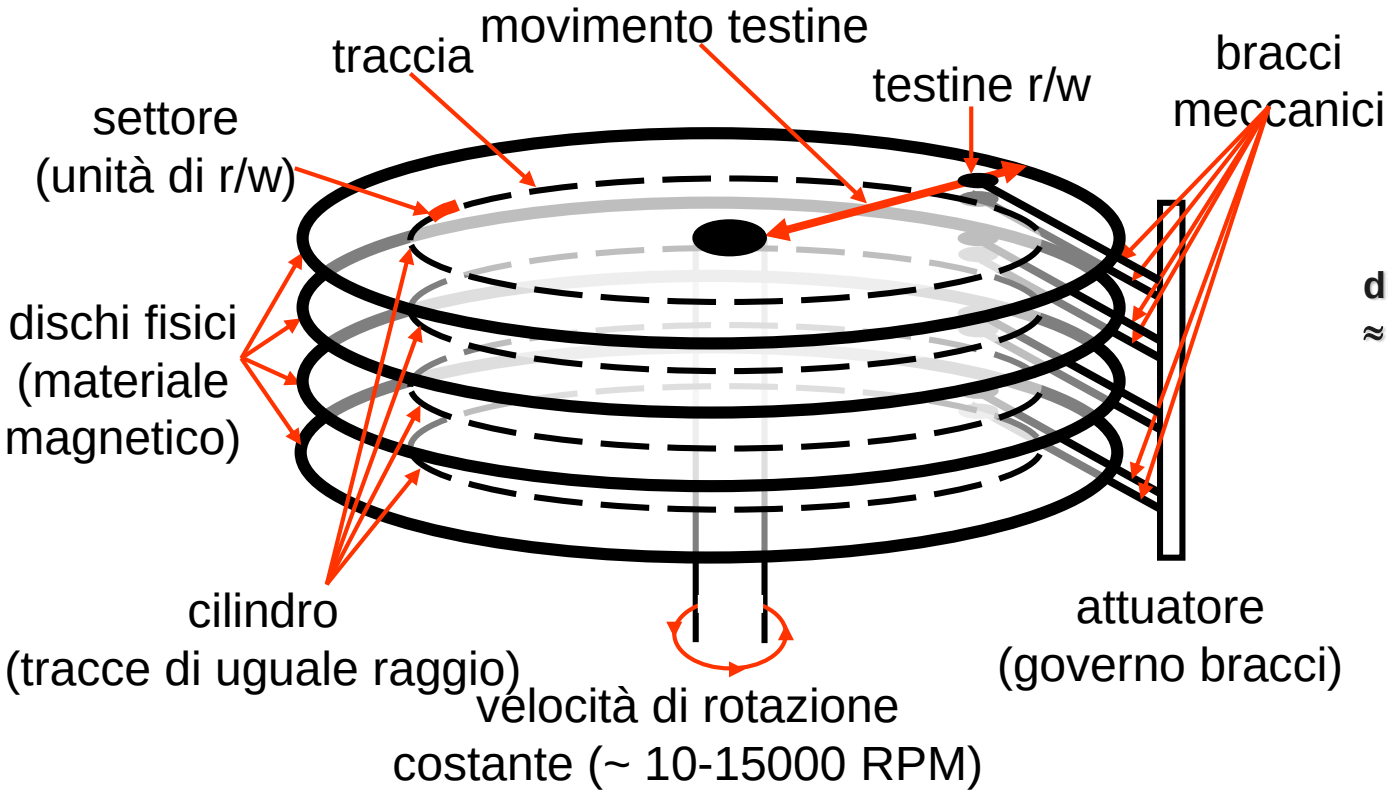
$p^*(i) = p(i) \times (1-m(i))$ *probabilità di trovare il dato al livello (i)*

$t^*(i)$ *tempo per raggiungere il livello (i) compreso*

tempo totale $= \Sigma\ p^*(i) \times t^*(i)$

architettura
dischi magnetici

dischi magnetici



distanza disco testina ≈ 10 nm

caratteristiche fisiche di un disco

- una pila di piatti con diametro da circa 1.3 a 5.25 inches - ciascuno con (al più) due facce
  - dischi più piccoli possono contenere meno dati ma possono ruotare più rapidamente, consumano meno energia e hanno minori distanze di seek
- ruotanti da 7200 a 15000 giri/minuto
- ogni faccia contiene 10000-50000 tracce concentriche
- divise in settori (100-500) tipicamente di 512 – 1024 byte: minima unità indirizzabile (sono stati usati anche blocchi a lunghezza variabile)
  - che contengono numero settore - gap- informazioni con sequenza di correzione errori - gap - numero prossimo settore e così via
- inizialmente ogni traccia conteneva lo stesso numero di settori (quindi di bit) ma con *ZRB* (*zone bit recording*) il numero dei settori varia da zona a zona, spostandosi verso l'esterno aumenta la capacità delle tracce e quindi anche la velocità di trasferimento
- distanza fra tracce contigue > separazione lineare bit sulla traccia

caratteristiche fisiche di un disco (cont.)

- le *testine* di lettura/scrittura
  - si muovono in modo solidale e si trovano contemporaneamente su tracce omologhe per ogni superficie
  - *cilindro*: insieme delle tracce di tutte le facce accessibili dalle testine posizionate in un determinato punto
  - *seek*: operazione meccanica di posizionamento della testina, la sua durata dipende dalle tracce attraversate, mediamente 3-14 ms
  - *latenza*: tempo necessario per iniziare ad operare sul blocco richiesto (una volta completato il seek), mediamente ≈ mezza rotazione

caratteristiche fisiche di un disco (cont.)

- *trasferimento*: 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)
    - ovviamente se il trasferimento avviene dalla cache non ha luogo né il seek né la latenza non essendo interessato nell'operazione alcun componente meccanico
    - in assenza di cache fra seek e trasferimento si possono perdere rotazioni, infatti:

    se ρ è l'utilizzo del canale di trasferimento = probabilità di non riconnessione allora:

    $\rho^k (1- \rho)$ è la probabilità di perdere k rotazioni

    $\Sigma\ k\ \rho^k (1- \rho) = \rho /(1- \rho)$ è il numero medio di rotazioni perse

caratteristiche fisiche di un disco (cont.)

- la densità dei dati dipende
  - dalla densità lineare della registrazione sulla traccia
  - dalla distanza fra tracce adiacenti
- il controller del disco generalmente decodifica il *logical block number* (LBN) in posizione fisica (CHS – *cylinder, head, sector*)
  - originariamente il valore CHS cresceva all'aumentare del LBN, questa mappatura permetteva di ridurre il numero di switch fra cilindri nel caso di operazioni sequenziali con LBN adiacenti; tecniche diverse vengono però usate dai dispositivi moderni, cioè l'indirizzo CHS non corrisponde alla geometria del disco che non è generalmente nota
  - se RAID o SAN o LUN (logical drive) LBN è trasformato diverse volte nell'indirizzo fisico
  - LBN o LBA (*logical block address - introdotto da architettura SCSI*) sono sinonimi

caratteristiche fisiche di un disco (cont.)

- cylinder/track *skew factors* rappresentano l'offset di inizio settore su tracce/cilindri adiacenti, servono a minimizzare il tempo di wait per latenza passando da una traccia o cilindro al successivo negli accessi sequenziali
- le velocità di rotazione hanno una certa tolleranza perciò dopo un certo numero di rotazioni (~100-200) una iniziale sincronia fra dischi diversi viene persa
- i dischi non sono perfetti: alcuni settori o tracce vengono sostituiti da *spare* (alla fabbricazione)

- un seek è composto da diverse fasi
  - *speedup:* il braccio accelera fino a quando raggiunge la massima velocità o la metà dello spazio di seek
  - *velocità costante:* (per seek di grande estensione)
  - *slowdown:* il braccio decelera fino a quando si trova nei pressi della traccia desiderata

caratteristiche fisiche di un disco (cont.)

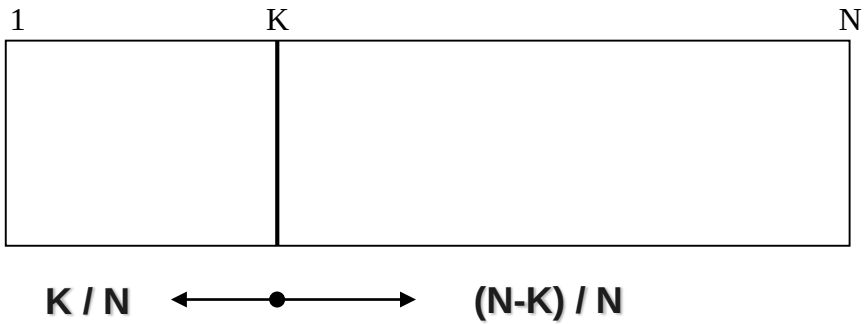
- *settle:* la testina si stabilizza per accedere al dato desiderato (il tempo è maggiore per write rispetto a read perché in questo caso il posizionamento deve essere più preciso)

- il tempo di seek non dipende linearmente dalla distanza
  - seek molto brevi (< 4 cyl) sono dominati dal tempo di posizionamento
  - seek brevi (< 200 – 400) hanno una durata proporzionale alla radice quadrata dell'intervallo, percorso ad accelerazione costante, (oltre al posizionamento)
  - seek molto lunghi hanno una durata proporzionale alla distanza, percorsa a velocità costante (più un overhead fisso)
- il tempo medio di seek dipende dal tipo di disco ma anche dal workload servito (sequenza di accessi)
  - se le richieste sono random e indipendenti l'estensione media di seek è 1/3 della dimensione totale (seek medio)
    - *ma il tempo medio di seek NON è il tempo per fare un seek medio* (tranne che nel caso di dipendenza lineare del tempo dallo spazio percorso)

Digressione: calcolo del “seek” medio

- se abbiamo una grandezza derivata ad esempio la durata dell’operazione di seek (posizionamento meccanico della testina sulla pista del dato)
  - tempo di seek = f(spazio attraversato)
- ha senso calcolare la media aritmetica (che conserva la linearità) come segue:
  - tempo medio di seek = f(spazio medio attraversato)

  solo se c’è una relazione lineare fra lo spazio attraversato e il tempo impiegato



Digressione: calcolo del "seek" medio (cont.)

Se partiamo da K e tutte le posizioni di arrivo sono ugualmente probabili lo spazio medio sarà:

$$\frac{K}{2}\cdot\frac{K}{N}+\frac{N-K}{2}\cdot\frac{N-K}{N}=\frac{1}{2N}\cdot\left(K^2+(N-K)^2\right)=\frac{1}{2N}\cdot\left(N^2-2NK+2K^2\right)$$

Se anche tutte le posizioni di partenza sono ugualmente probabili e N abbastanza grande lo *spazio* medio di seek sarà:

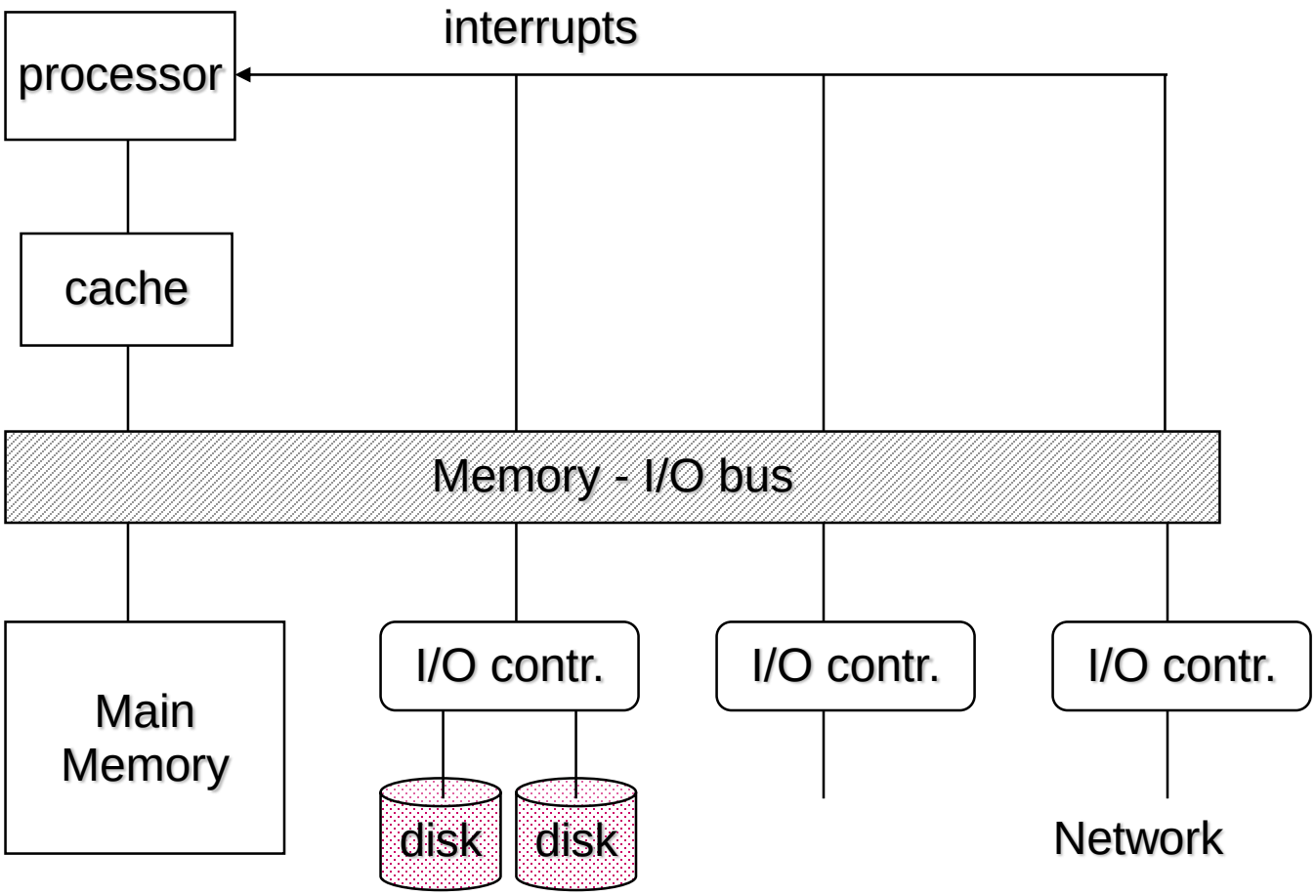
$$\frac{1}{N}\cdot\sum_{K=0}^{N}\frac{1}{2N}\cdot\left(N^2-2NK+2K^2\right)=\frac{N+1}{2}-\frac{1}{N}\cdot\sum K+\frac{1}{N^2}\cdot\sum K^2=\frac{1}{N^2}\cdot\sum K^2$$

$$=\frac{1}{N^2}\cdot\frac{1}{3}\left[(N+1)^3-(N+1)-\frac{3}{2}\cdot N(N+1)\right]=\frac{(N+1)\left(N+\frac{1}{2}\right)}{3N}\to\frac{N}{3}+\frac{1}{2}\approx\frac{N}{3}$$

ma il *tempo* medio di seek deve essere calcolato come media dei tempi cioè:
tempo medio di seek = Σ ti × pi

somma dei quadrati degli interi consecutivi da 1 a N

sistemi di I/O



bus

- *bus*: link di comunicazione condiviso che usa un insieme di *cavi* per connettere sottosistemi multipli (circuiti di indirizzo, dati e controllo)

- *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

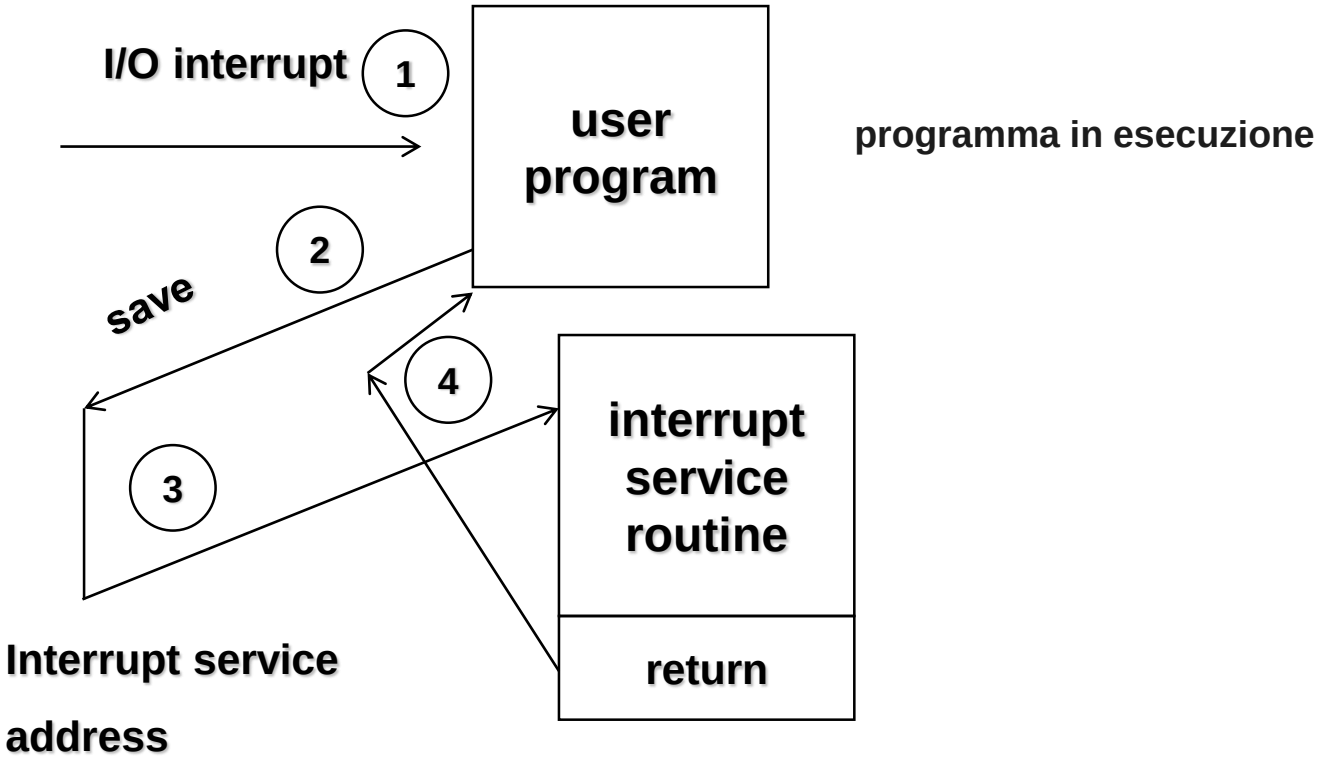
- *svantaggi*:
  - collo di bottiglia delle comunicazioni, la larghezza di banda limita il massimo throughput I/O
  - lunghezza del bus
  - numero di dispositivi

- *prospettive*:
  - switch di interconnessioni seriali ad alta velocità

bus (cont.)

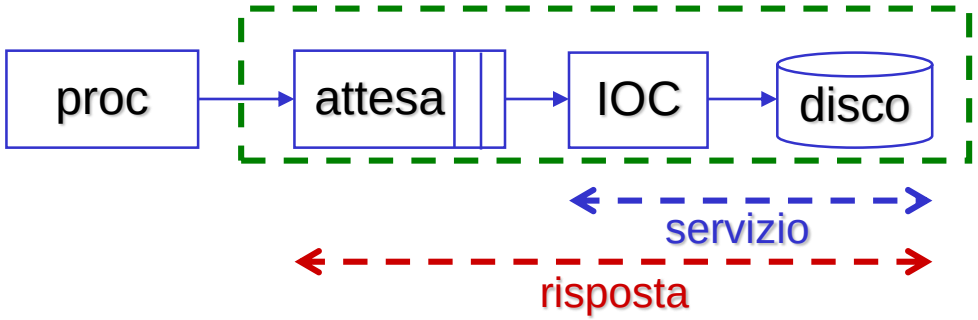
- tipi di bus
  - *processore - memoria* (corti e ad alta velocità)
  - di *I/O* (lunghi con ampio range di banda)
  - *backplane* (permette la coesistenza di memoria e I/O)
  - sono organizzati in modo gerarchico
- protocollo di comunicazione
  - *sincrono* (generalmente usato dal bus di memoria); i dispositivi devono avere lo stesso ritmo di clock
  - *asincrono* (usato dai bus di I/O); il coordinamento della trasmissione è gestito da un protocollo di *handshaking*; richiede linee addizionali per i segnali di controllo
- trasferimento
  - *parallelo* (lunghezza limitata del collegamento)
  - *seriale*

Interrupt Driven Data Transfer (schema)



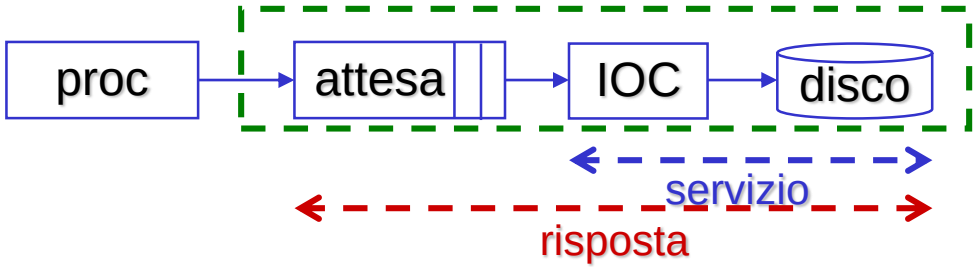
prestazioni dischi
(calcoli approssimati)

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 (≈ ms)
  - tempo di latenza: tempo richiesto affinché il settore passi sotto le testine (≈ ms, ½ giro)
  - tempo di trasferimento dei dati : dipende da velocità di rotazione, densità di registrazione, distanza della testina dal centro del disco (≈ MB/sec)
  - *overhead controller:* gestione trasferimento dati da buffer locale e invio comunicazioni di controllo (interrupt)
  - tempo di switch/attivazione testina (generalmente inglobato nel seek)

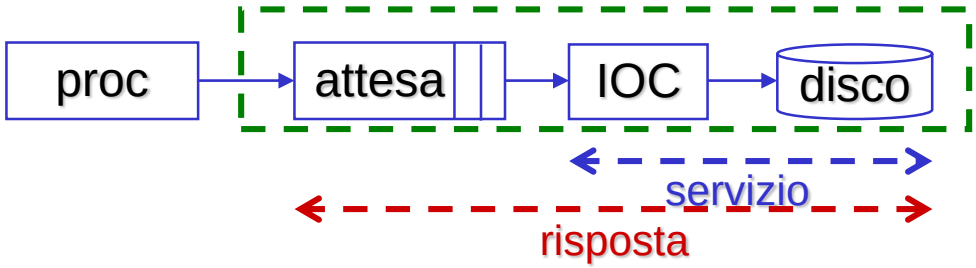
metriche di prestazione di un disco (cont.)



- 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 arrivo delle richieste e loro distribuzione
  - *a parità delle altre condizioni una maggiore variabilità determina un tempo di risposta mediamente più grande*

metriche di prestazione di un disco (cont.)

calcolo delle statistiche
risultanti da composizione
o somma di diversi fenomeni

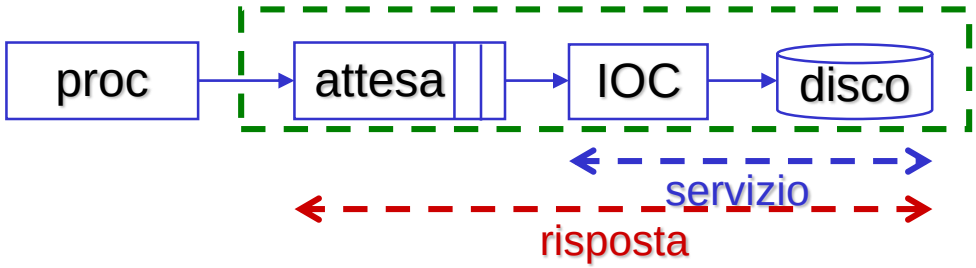


- *M1: momento del 1° ordine (media); M2: momento del 2° ordine*
- *il tempo di servizio Ti* varia da richiesta a richiesta (*composizione*):
  - *Tempo medio* ti = M1 =(f1×T1+f2×T2+...+fn×Tn)

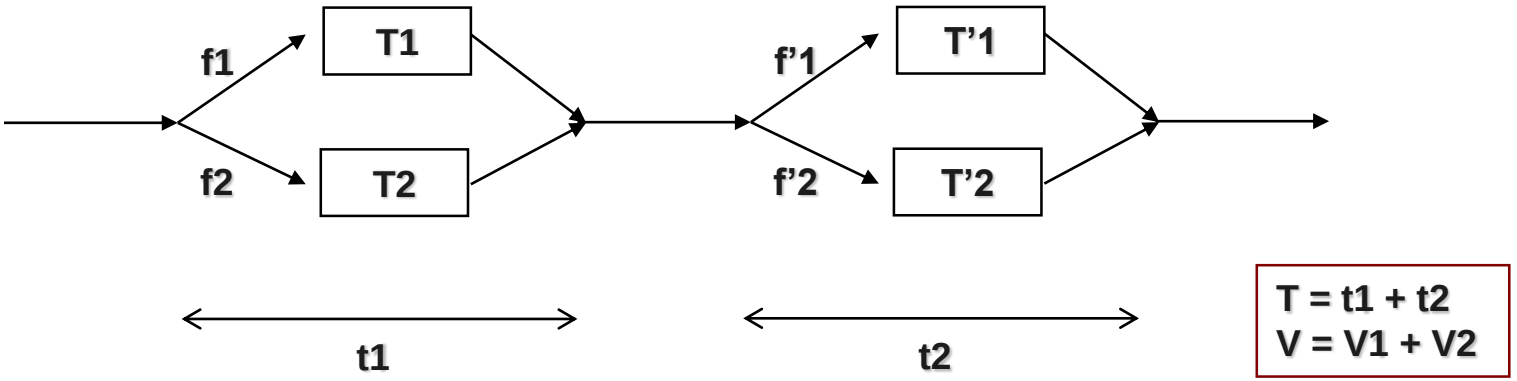
  (f1+f2+...fn=1)
  - *Varianza* Vi =(f1×$T1^2$+f2×$T2^2$+...+fn×Tn^2) - $M1^2$ = M2 - $M1^2$
  - C^2 = quadrato del coefficiente di variazione = Varianza / $M1^2$
- *un tempo medio di servizio T* è la *somma* di tempi medi indipendenti:
  - T = t1+t2+…tm (seek+rotazione+trasferimento+overhead)
  - V = V1+V2+...Vm

metriche di prestazione di un disco (cont.)

composizione e somma di diversi fenomeni: calcolo delle statistiche



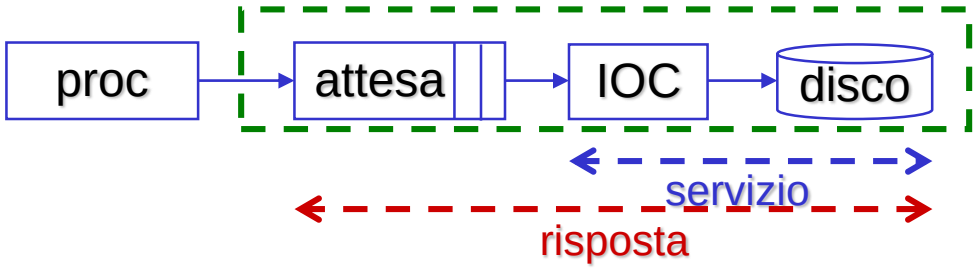
- *algoritmo della composizione / somma dei tempi*



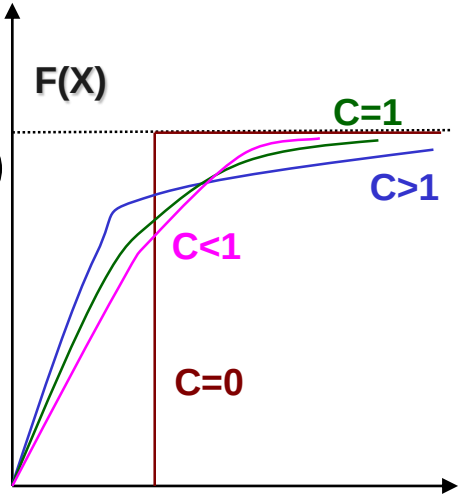
$t1 = f1 \times T1 + f2 \times T2$
$V1 = (f1 \times T1^2 + f2 \times T2^2) - t1^2$

$t2 = f'1 \times T'1 + f'2 \times T'2$
$V2 = (f'1 \times T'1^2 + f'2 \times T'2^2) - t2^2$

metriche di prestazione di un disco (cont.)



- il *coefficiente di variazione* C = std.dev/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



esercizio 4: tempo medio di servizio di una op. di I/O

- lettura/scrittura di un settore di 512 Byte = 0.5 KB,
  - velocità di rotazione: 10000 RPM (rotazioni per minuto)
  - velocità di trasferimento dati: 50 MB/sec
  - seek medio: 6ms
  - overhead controller: 0.2ms
  - latenza: (60s/min)x1000/(2x10000giri/min)=3.0ms (tempo per compiere mezza rotazione)
  - trasferimento: (0.5KB)x1000 / (50x1024KB/s) = 0.01ms
- tempo totale di servizio di I/O = 6ms+ 3ms+ 0.01ms+ 0.2ms = 9.21ms

seek **latenza** **trasferimento** **controller**

esercizio 5: effetti della località degli accessi

- località: effetto della –
- riprendendo i dati dell'esercizio precedente:
  - località dei dati: seek ha luogo solo nel 25% delle operazioni

$(6.0 \times 0.25) + (0.5 \times 60 \times 10^{3} / 10000) + (0.5\ KB / 50MB \times 2^{10}) + 0.2 =$

$1.5 + 3.0 + 0.01 + 0.2 = 4.71\ ms$

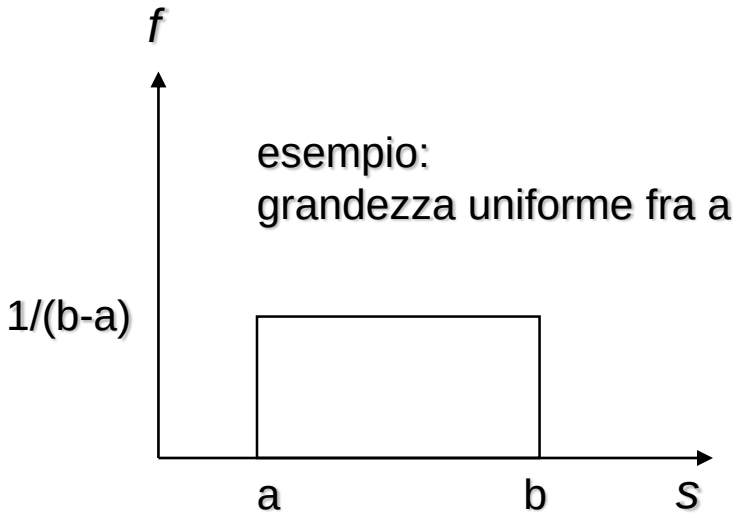
  - tempo medio = $(0.25 \times 6) + 3 + 0.01 + 0.2 = 4.71$ ms

seek **latenza** **controller** **trasferimento**

  - ipotesi: il tempo di seek ha distribuzione uniforme fra 0.4 e 11.6 ms

calcolo del tempo di servizio di una op. di I/O

- *tempo medio s = seek + latenza + trasferimento + controller*
- *ipotesi:*
- seek: uniforme fra un valore minimo e un massimo
- latenza di rotazione: uniforme fra 0 e tempo di rotazione completa
- trasferimento: costante
- controller: costante



media M1 = (b+a) / 2

2do Momento M2 = $(b^2 + ab + a^2) / 3$

Varianza V = $(b-a)^2 / 12 = M2 - M1^2$

$$M2 = \int_a^b x^2 \frac{1}{b-a} dx$$

esercizio 6: Varianza del tempo di servizio di I/O

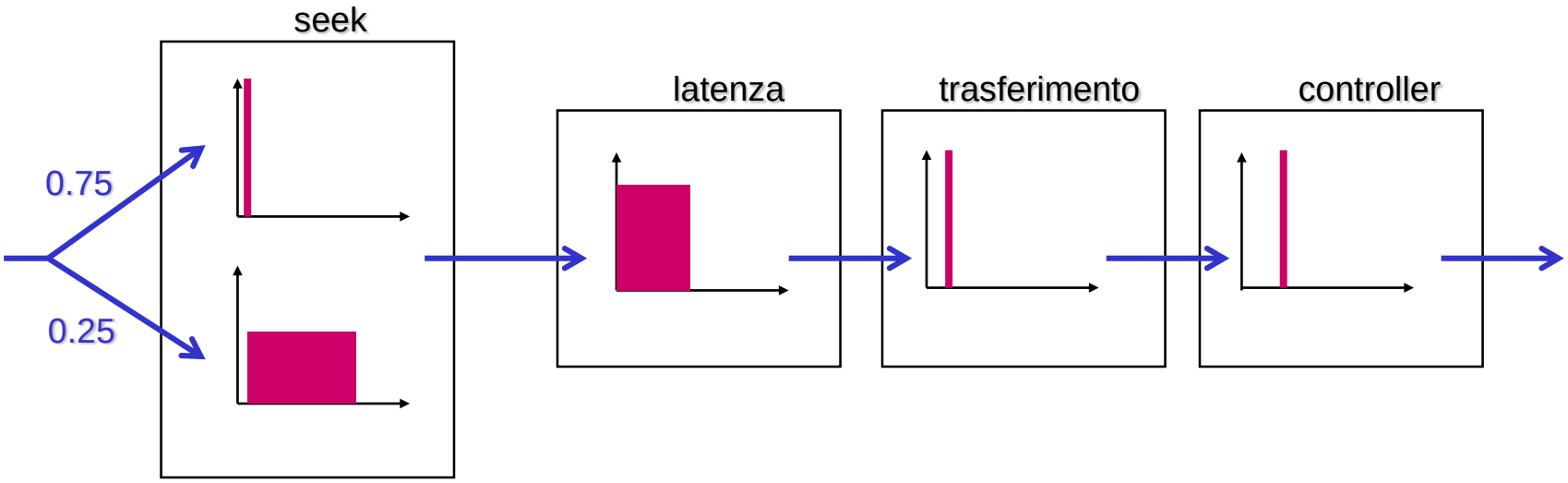
- V(tempo di seek): **(seek ≠ 0 nel 25% dei casi)**
  - $M2 = 0.25 \times (11.6^2 + 11.6 \times 0.4 + 0.4^2) / 3 = 0.25 \times 46.45$
  - $V = M2 - M1^2 = 11.61 - (0.25 \times 6)^2 = 9.36$
- V(tempo di latenza):
  - $V = 6^2 / 12 = 3$
- V(tempo di trasferimento):
  - V = 0
- V(tempo di controller):
  - V = 0
- *Varianza totale* = 9.36 + 3 + 0 + 0 = 12.36

i dati provengono dall'esercizio 5

seek latenza controller trasferimento

- *coeff. di variazione* = $\sqrt{(12.36)}/4.71 = 0.747$
- $0.747 < 1 \Rightarrow$ distribuzione ipo-esponenziale

esercizio 6: riassunto grafico



t1 = 0 ; V1 = 0
t2 = 6 ; V2 = 10.45

T(seek) = 1.5
V(seek) = 9.36
C = 2.04

T(latenza) = 3
V(latenza) = 3
C = 0.58

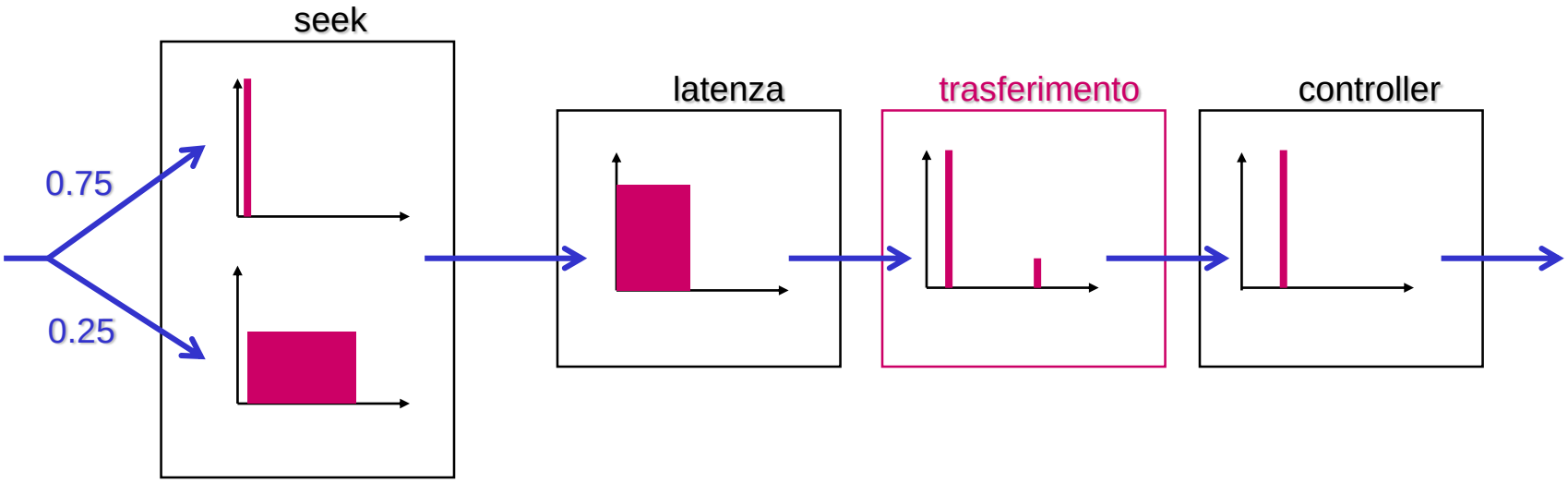
T(trasf.) = 0.01
V(trasf.) = 0
C = 0

T(contr.) = 0.2
V(contr.) = 0
C = 0

il fenomeno complessivo ha un tempo medio s = Σ T = 4.71

e varianza = Σ V = 12.36 ; C = 0.747

esercizio 7: variazione sui tempi di trasferimento



t1 = 0 ; V1 = 0
t2 = 6 ; V2 = 10.45

T(seek) = 1.5
V(seek) = 9.36
C = 2.04

T(latenza) = 3
V(latenza) = 3
C = 0.58

T(trasf.) = 0.609
V(trasf.) = 3.23
C = 2.95

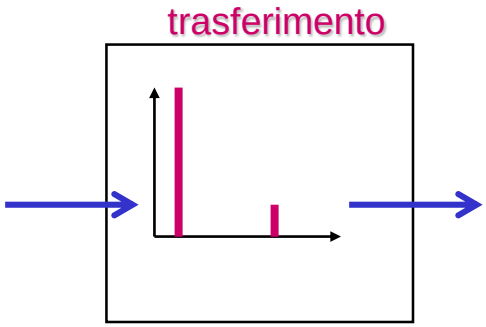
T(contr.) = 0.2
V(contr.) = 0
C = 0

il 90% delle I/O trasferisce 512 byte,
il 10% una intera traccia

il fenomeno complessivo ha un tempo medio s = Σ T = 5.31

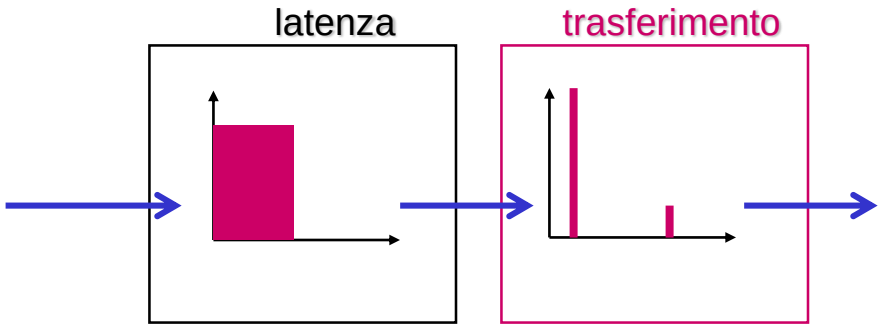
e varianza = Σ V = 15.59; C = 0.744

esercizi 6 e 7: osservazione sulle varianze



Più un fenomeno assume valori diversi
più il coefficiente di variazione tende a crescere

esempio: tempo di trasferimento
t1 = 0.01; **c = 0** nel 90% dei casi
t2 = 6; **c = 0** nel 10% dei casi
t(trasf totale) = 0.609; **c(trasf totale) = 2.95**



se un fenomeno è la somma di più
fenomeni diversi,
il coefficiente di variazione totale è minore
della media dei singoli coefficienti

esempio: tempo di latenza + trasferimento
t(latenza) = 3; **c(latenza) = 0.58**
t(trasf totale) = 0.609; **c(trasf totale) = 2.95**
t(latenza + trasferimento) = 3.609
c(latenza + trasferimento) = 0.69

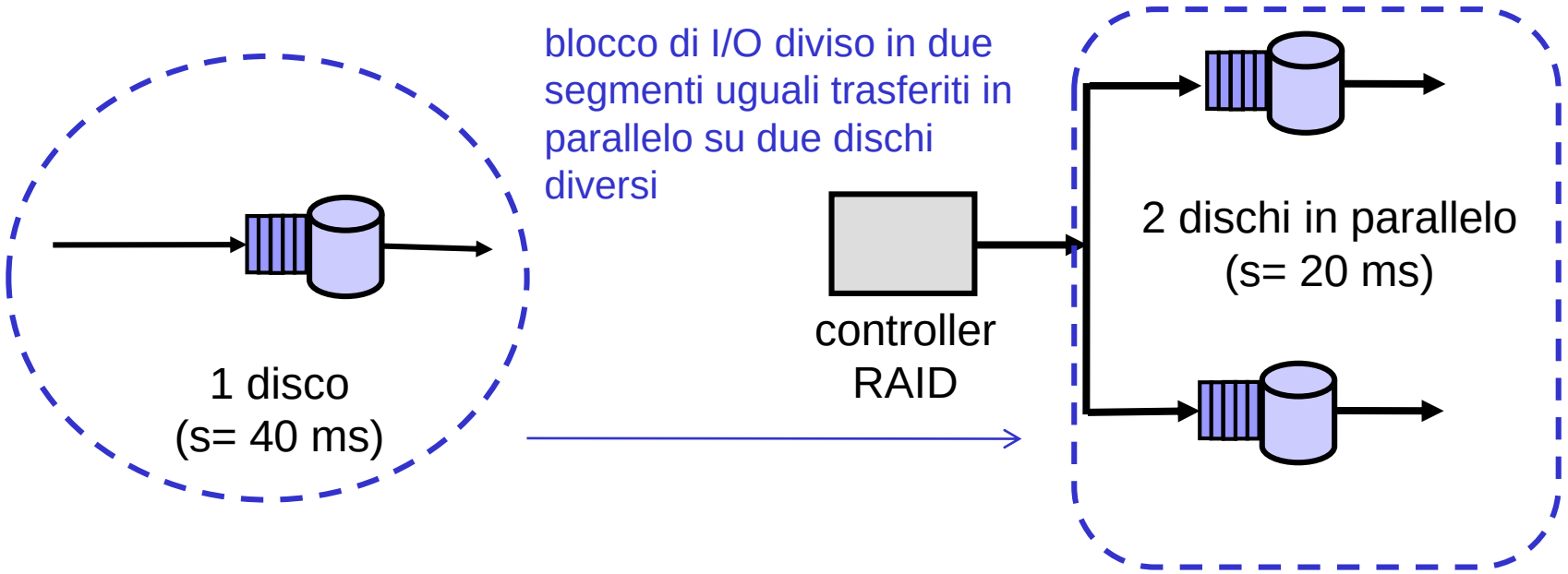
c = σ / media

esercizio 8: effetto di una allocazione dati “frammentata”

- trasferimento di un file (size = 1MB)
- 1° caso:
  - 1 seek iniziale: 6 ms
  - 1 latenza: 3 ms
  - 1 trasferimento totale 1 MB: 1/50 × 1000 = 20 ms
  - tempo totale: 29 ms
- 2° caso (10 blocchi da 1/10 MB distribuiti “male” sul disco), per ciascun blocco:
  - 1 seek: 6 ms
  - 1 latenza: 3 ms
  - 1 trasferimento parziale: 2 ms
  - perciò, tempo totale: (6 + 3 + 2) × 10 = 110 ms

(non si è tenuto conto dei tempi di overhead del controller)

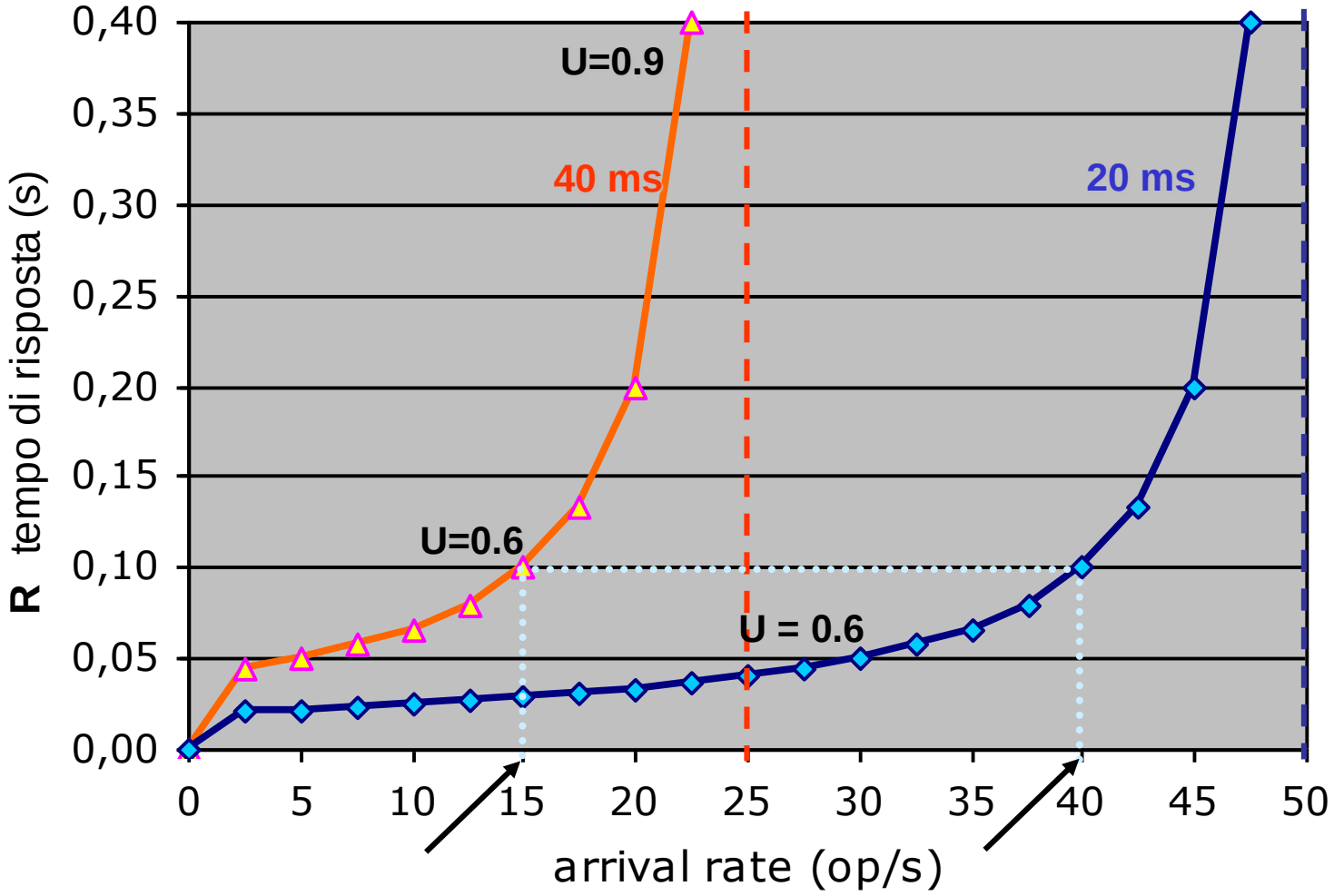
esercizio 9: 2 dischi in parallelo (modello M/M/1)



- si ipotizza che le operazioni vengano effettuate in parallelo sui due dischi con tempi di servizio pari alla metà di quello originario
- 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 (naturalmente il costo è doppio: 2 dischi invece di 1)

esercizio 9: 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

- considerazioni:
  - 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;
  - se si vuole studiare l'effetto della *cache* per un file system le I/O vanno tracciate a livello logico;
  - ogni tecnica è dominante per qualche effetto, deve essere valutata isolatamente dalle altre e ha un comportamento ottimale se implementata ad un certo livello della gerarchia di storage

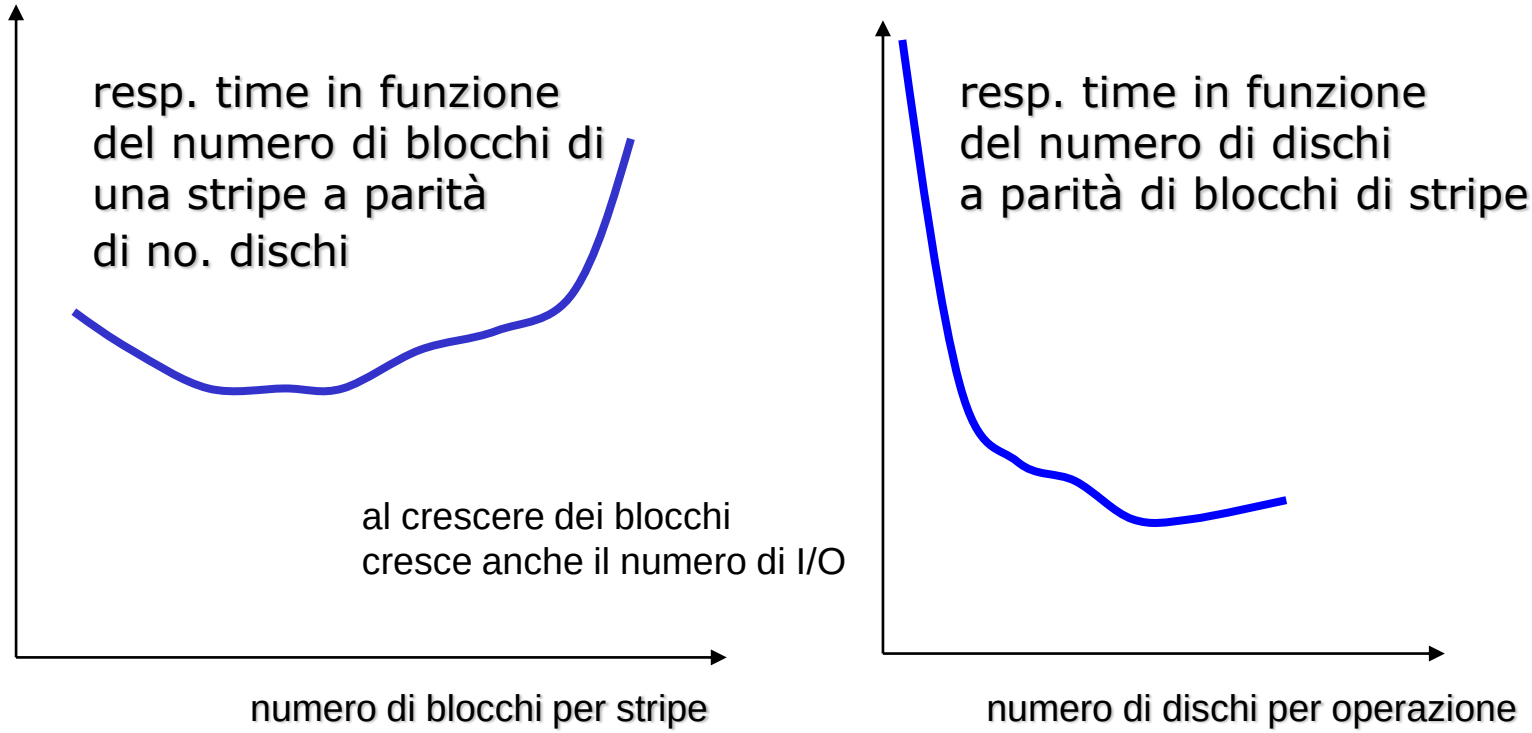
tecniche per migliorare le prestazioni I/O (cont.)

- per studiare sistemi fisici bisogna ottenere la traccia delle I/O fisiche, non tutte vengono intercettate (es.. I/O sul page system)

- due metriche importanti sono ***response time*** e ***throughput***.
- il reciproco del *service time* è una stima (ottimistica) del *troughput massimo* ottenibile (con utilizzo = 1)
- l'efficienza delle tecniche di miglioramento dipende in modo significativo dall'organizzazione dei dati e dalla tipologia di workload

striping: 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.



striping: ripartizione uniforme delle operazioni

- un risultato importante, dal punto di vista delle prestazioni, della tecnica di *striping* (che in prima approssimazione può essere pensata come la suddivisione di una singola operazione in diverse eseguite in parallelo) è il fatto che gli accessi si ripartiscono automaticamente in modo uniforme fra i dischi interessati

- in condizioni stazionarie una distribuzione omogenea delle richieste fra dispositivi (di identiche caratteristiche) è quella che garantisce il minore tempo medio di risposta (si veda l'esercizio alla pagina seguente)

- se non si usano tecniche di striping gli accessi tendono a distribuirsi con legge di potenza:
  - se ordiniamo i dischi in ordine di numero di accessi e R(n) è il numero di accessi fatti sui dischi dalla posizione n in poi, allora
  - $R(n) \approx c\, n^{-\alpha}$ perciò:
  - $R(2n) = R(n) / 2^{\alpha}$ $\quad (\alpha > 1)$

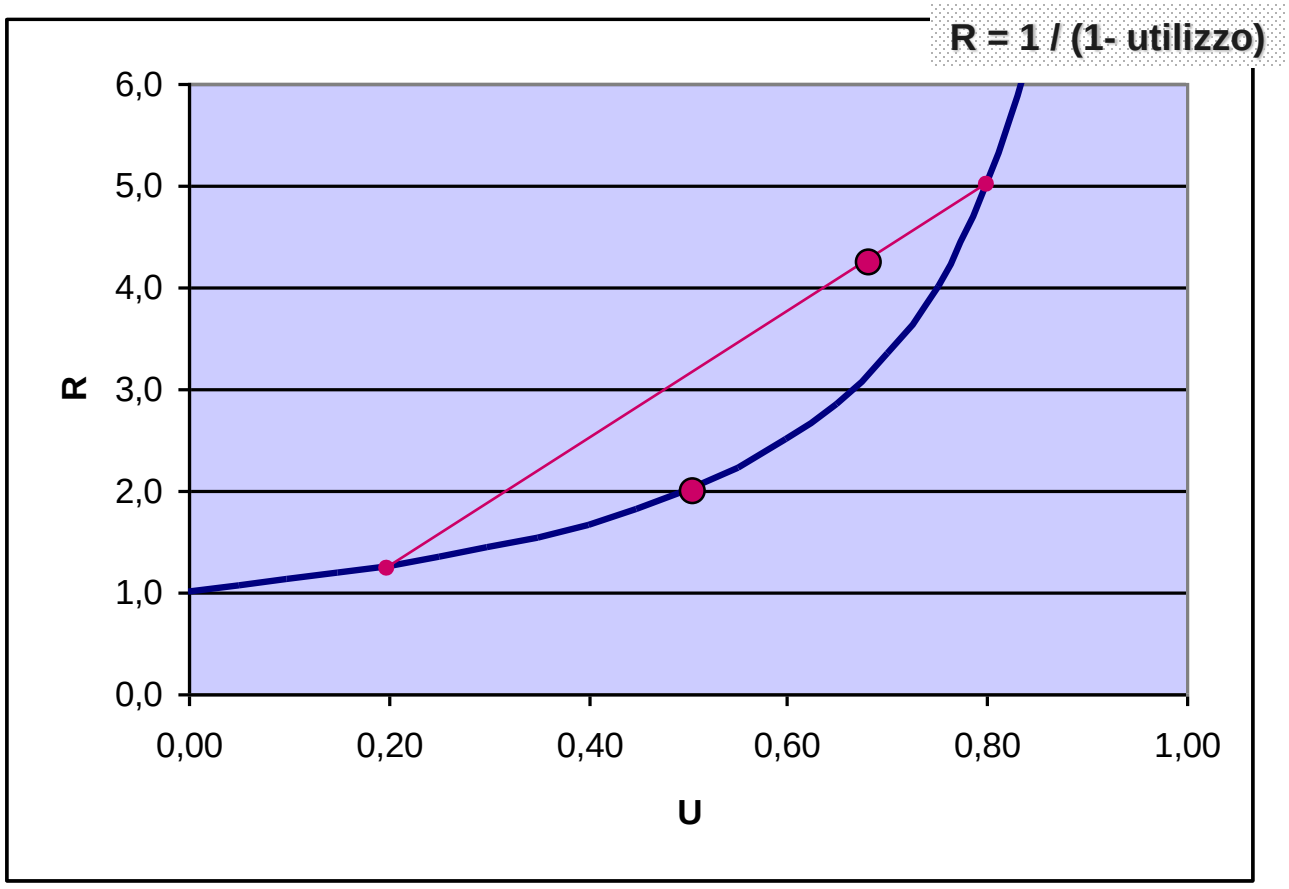
esercizio 10: ripartizione delle operazioni

- un certo carico può ripartirsi su due dischi secondo due distinte modalità:
  - **20% e 80%** oppure **50% e 50%**
  - ipotizziamo che se tutto il carico avesse a disposizione un solo disco, questo sarebbe utilizzato al 100%, perciò la ripartizione del carico può essere usata anche come utilizzo dei dischi stessi
  - i tempi di risposta R (attesa + servizio) in funzione del carico sono riportati nel grafico successivo (approssimazione M/M/1)

    - nel primo caso abbiamo **R(disco1) = 1.25; R(disco2) = 5**
    - nel secondo **R(disco1) = R(disco2) = 2**
- il tempo medio di risposta vale, nelle due ipotesi, rispettivamente :
  - $0.2 \times 1.25 + 0.8 \times 5 =$ **4.25** (ripartizione disomogenea)
  - $0.5 \times 2 + 0.5 \times 2 =$ **2** (ripartizione omogenea)

esercizio 10: ripartizione delle operazioni (cont.)

- due dischi utilizzati rispettivamente al **20** e **80** percento hanno un tempo medio di risposta di **4.25**
- se invece sono utilizzati al **50** percento hanno entrambi un tempo di **2.0**



read caching

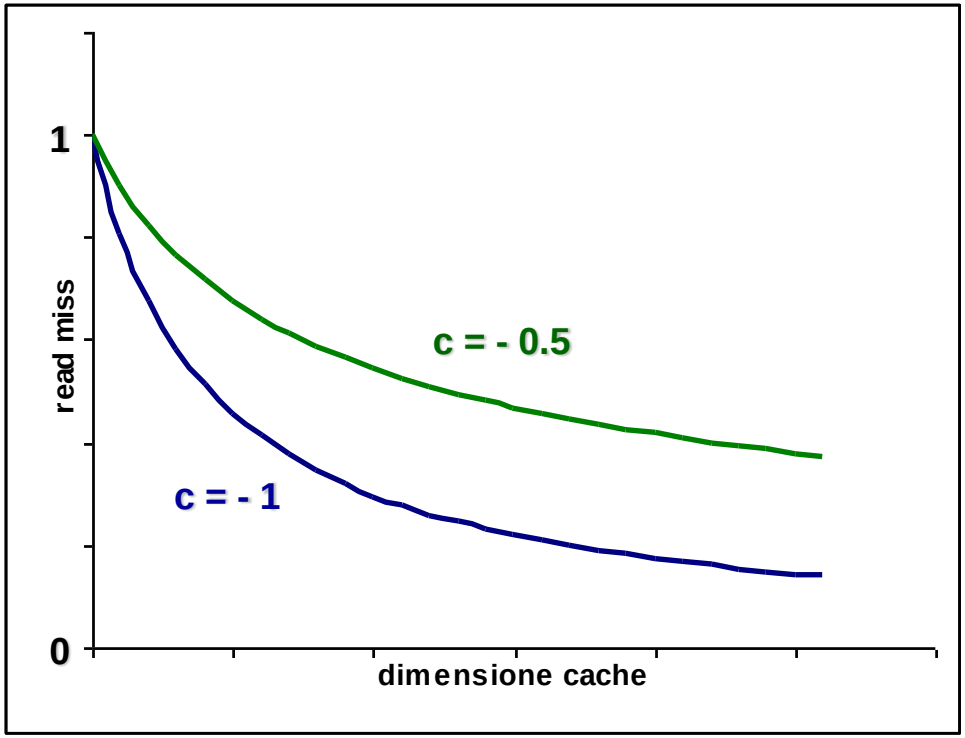
- tecnica che consiste nel tenere temporaneamente in una memoria più veloce (*cache*) quei dati che è probabile vengano usati in seguito.
  - dati: blocchi di storage (generalmente di 4KB);
  - cache: DRAM (efficiente se > 32MB) in memoria;
  - anche molti dischi hanno una loro unità cache (2-4-8 MB) che serve soprattutto come buffer di *prefetching (vedi più avanti);*
  - metodo di alimentazione *least-recently-used* (LRU) giustificato dalla "locality" dei dati.

- *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 (8 MB = 4% 200 MB)

read caching (cont.)

- l'analisi di dati sperimentali mostra un *miss ratio* che segue la relazione:
- $f(x) = a(x-b)^c$ *(a,b,c costanti sperimentali , c ≅ -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 il *caching* a livello fisico è più efficiente di quello ottenuto a livello logico).



read miss = funzione della memoria che agisce da cache

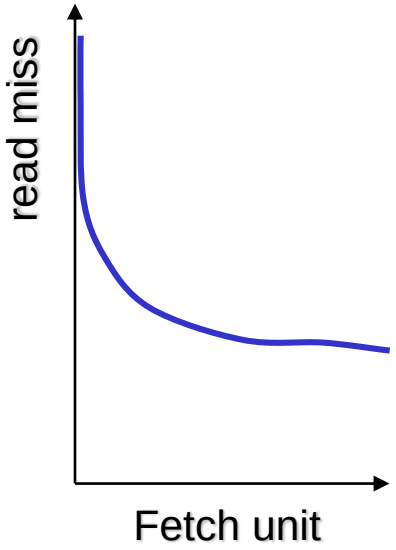
prefetching

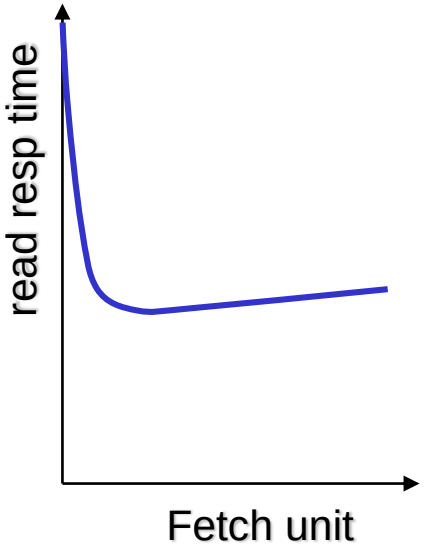
- precarica in memoria i blocchi che si presume vengano usati a breve, la sua efficienza dipende da:
  - accuratezza della previsione;
  - tempestività (perché l'operazione deve essere completata prima che i blocchi siano richiesti);
  - costi: risorse consumate (memoria e altri componenti)

- la maggior parte dei carichi presenta una certa sequenzialità negli accessi
  - in occasione di un *cache miss* si può realizzare *prefetching* sequenziale;
  - molti accessi sono trasformati in un singola operazione

prefetching (cont.)

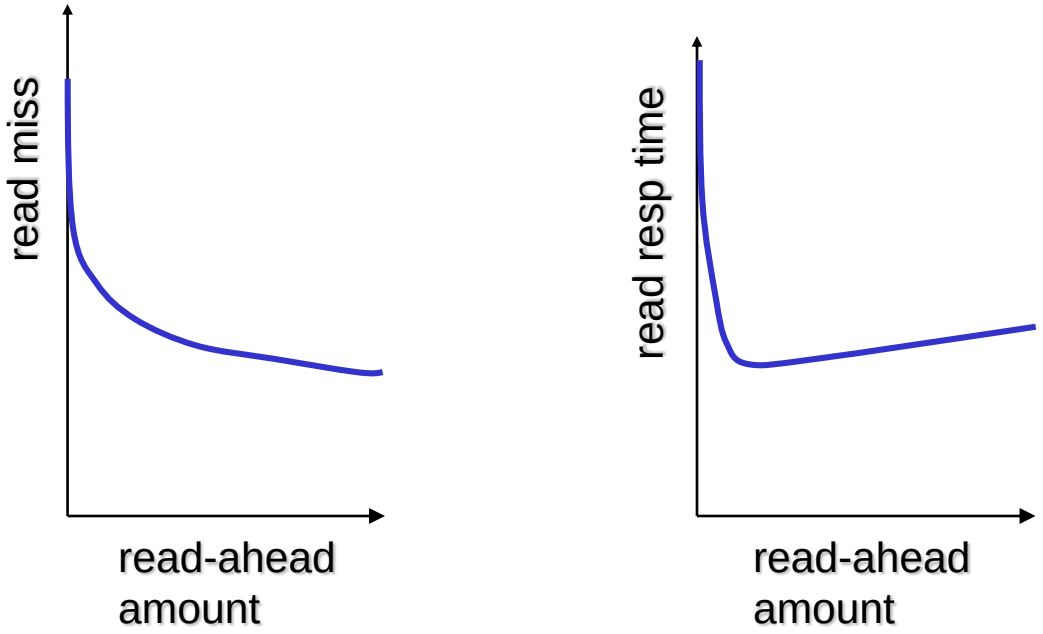
- *Large fetch unit*
  - tecnica che esegue il caricamento di un certo numero di blocchi che precedono e che seguono quello effettivamente richiesto;
  - *response time penality* bisogna attendere che il trasferimento sia completato, in alternativa si possono eseguire due operazioni: fino al blocco “target”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





prefetching (cont.)

- *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 (cont.)

- *Preemptible read ahead:* semplice forma di *prefetch* che usa solo risorse che sarebbero altrimenti non usate *(idle)*
  - la richiesta di prefetching è divisa in tante sotto-richieste
  - la sequenza è interrotta quando arriva una nuova richiesta
  - si garantiscono buone prestazioni anche al crescere della domanda (numero di blocchi)
- *approccio ibrido:*
  - si inizia a leggere la traccia su cui si è posizionati,
  - si prosegue di 32 KB oltre i blocchi richiesti,
  - se non ci sono nuove richieste, si prosegue fino a 128 KB.
  - i miglioramenti delle prestazioni, nel caso dei server, sono almeno del 50% rispetto ai sistemi senza prefetching. (dipende dalla tipologia dei dati)

write buffering

- mantiene temporaneamente in memoria i blocchi da scrivere sullo 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)

- rischio di perdita di dati
  - il *write buffer* è eseguito su memoria non volatile (NVS) periodicamente il contenuto dei buffer è trasferito su disco *(destaging)*

- maggiore efficienza e riduzione del numero di operazioni fisiche
  - una singola write fisica combina operazioni multiple sulla stessa posizione;
  - write multiple consecutive sono combinate in una operazione su indirizzi contigui

write buffering (cont.)

- si tenta di ridurre il numero di write fisiche operando il *destage* dei blocchi che è meno probabile vengano riscritti
  - 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 (cont.)

- 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)