

www.dis.uniroma1.it/~midlab

Sistemi Operativi II

Corso di Laurea in Ingegneria Informatica

Prof. Roberto Baldoni

Complementi: Buffer I/O

Gestione dei buffer e I/O scheduling:

1. Richiami sulle tecniche di I/O
2. Gestione dei buffer
3. Schedulazione del disco

Tipologia dei dispositivi

Leggibili dall'uomo

- terminali video
- tastiera mouse
- stampanti

Leggibili dalla macchina

- dischi
- nastri
- controller ed attuatori

Di comunicazione

- modem
- schede di rete

Diversificati in base a

- velocità di trasferimento
- complessità del controllo
- unità di trasferimento (blocco)
- rappresentazione dei dati (codifiche)
- condizioni di errore e relativo rilevamento

Tecniche per l'I/O

I/O programmato

- il processore rimane in attesa attiva fino al completamento dell'interazione con lo specifico dispositivo

I/O interrupt-driven

- il processore impartisce un comando al dispositivo di I/O
- torna poi ad eseguire le istruzioni successive e viene interrotto dal dispositivo quando esso ha completato il lavoro richiesto

DMA

- un dispositivo DMA controlla lo scambio di dati tra memoria e dispositivi di I/O
- il dispositivo DMA interrompe il processore nel momento in cui operazioni di I/O precedentemente richieste sono completate

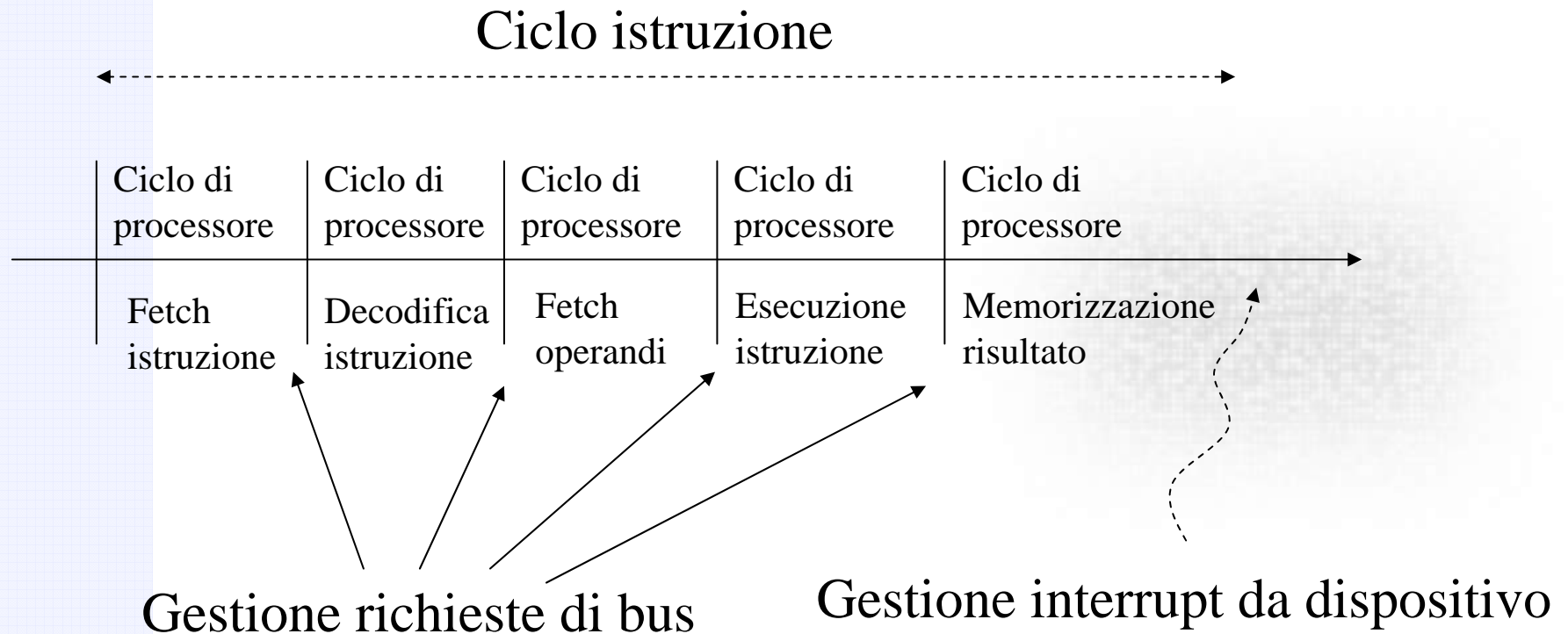
Evoluzione storica

| | Senza interrupt | Con interrupt |
|---|-----------------|----------------------|
| Trasferimento di I/O tramite processore | I/O programmato | I/O interrupt-driven |
| Trasferimento di I/O tramite DMA | | DMA |

1. Il processore controlla direttamente la periferica
2. Aggiunta di un controller che disaccoppia il processore dalla periferica
 1. Programmed I/O
 2. Interrupts (incrementa l'efficienza)
3. Il controller ha un controllo diretto della memoria via DMA (controllo del processore all'inizio ed alla fine delle operazioni)
4. Il controller diventa un vero e proprio micro processore con il proprio set di istruzioni (interfaccia con il processore principale in memoria principale)
5. Il processore di I/O ha una propria memoria interna

DMA ed interrupt

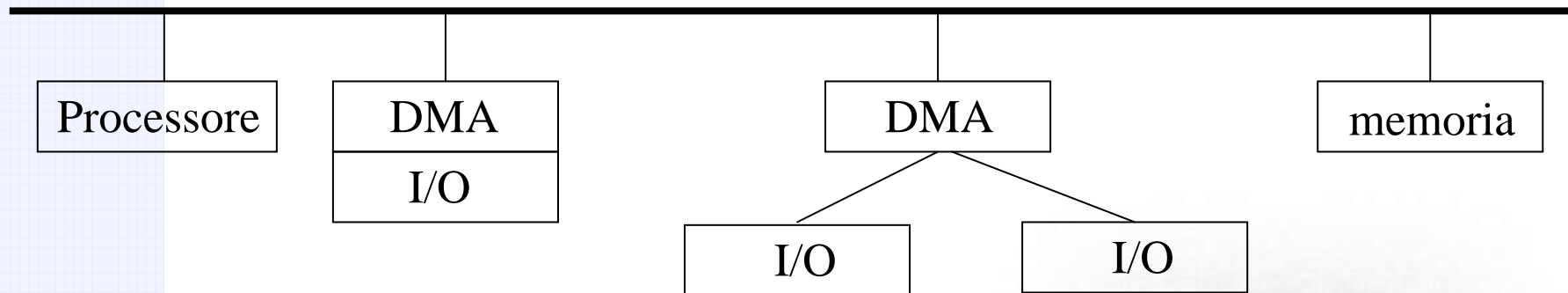
- le interruzioni relative alla richiesta di bus vengono gestite ad ogni ciclo di processore
- le interruzioni dei dispositivi (compreso il DMA) vengono gestite tra una istruzione macchina e l'altra



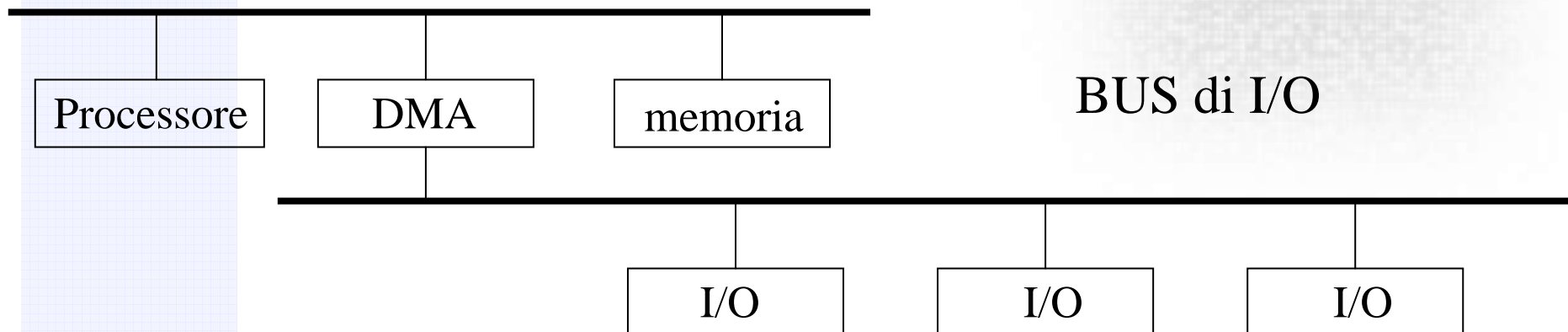
Configurazioni



BUS singolo - DMA separato



BUS singolo - DMA e I/O integrati



BUS di I/O

Progettazione della funzione di I/O: obiettivi

Efficienza

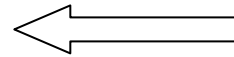
- l'interazione con i dispositivi è tipicamente il collo di bottiglia
- lo swapping tende ad aumentare il throughput tramite incremento del livello di multiprogrammazione
- lo swapping richiede però I/O efficiente per essere applicabile
- necessità di algoritmi efficienti per la gestione dell'I/O su disco

Generalità

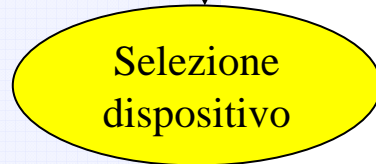
- uniformità di trattamento dei dispositivi da parte delle applicazioni
- fornitura di servizi di I/O con punti di accesso (interfacce) standard indipendentemente dalla tipologia di dispositivo
- approccio gerarchico alla progettazione, che nasconda i dettagli di più basso livello

Un modello di organizzazione

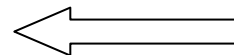
applicazioni



I/O su terminale, rete,
sistema di archiviazione (file system)



Algoritmi specifici in funzione delle
caratteristiche del dispositivo



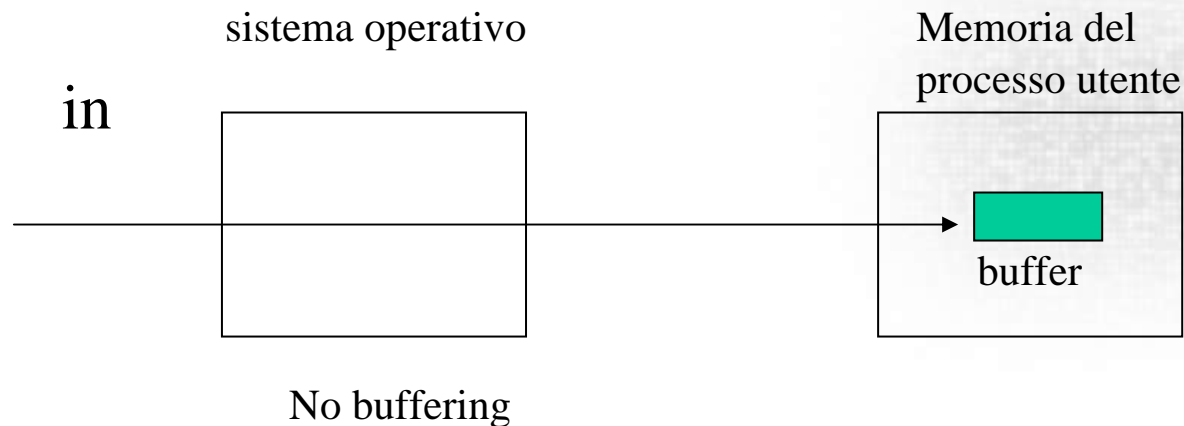
Dispositivo locale,
porta di comunicazione (rete)

Necessità di bufferizzazione

I/O effettuato direttamente sulla memoria riservata ai processi può provocare

1. Sottoutilizzo dei dispositivi e della CPU. Nel caso in cui l'area di memoria destinata per l'I/O non è "swappabile"
2. Deadlock nel caso in cui l'area di memoria destinata all'I/O sia swappabile (processo bloccato in attesa dell'I/O – I/O bloccato in attesa che il processo venga riportato in memoria)

Dispositivi
di I/O



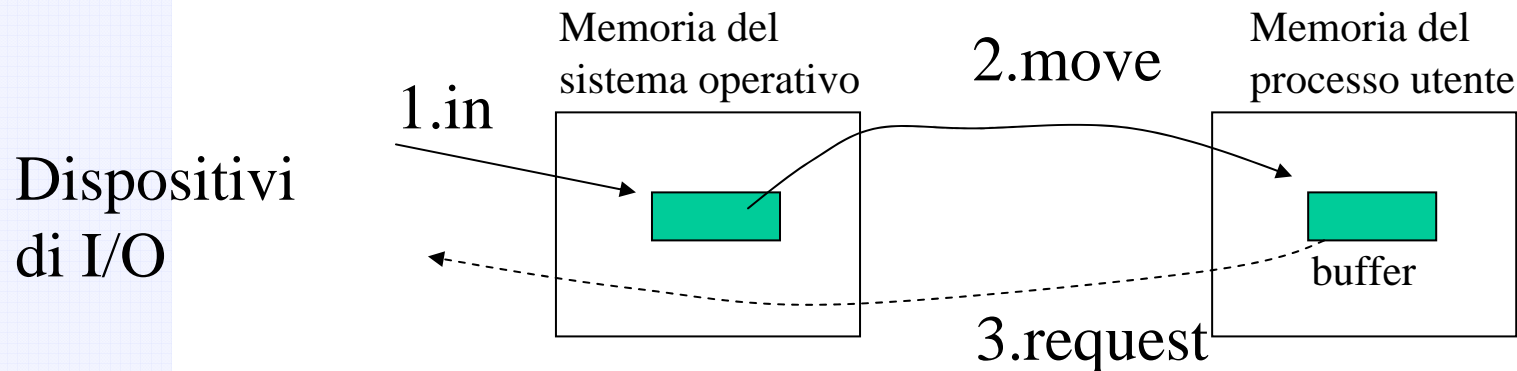
Necessità di bufferizzazione

- l'I/O viene quindi tipicamente effettuato su aree di memoria riservate al sistema operativo
- questo introduce la necessità di gestire buffer destinati alle operazioni di I/O

Orientamento

- a **blocchi**: bufferizzazione e trasferimento di blocchi di bytes
- a **flusso** (stream): bufferizzazione e trasferimento di singoli bytes

Singolo buffer



- viene assegnato un singolo buffer per la gestione delle operazioni di I/O
- in fase di input, immediatamente dopo la copia del contenuto del buffer in spazio utente, viene effettuata la lettura del blocco successivo (**lettura in avanti – input anticipato**)
- il sistema operativo comincia a complicarsi poiché' si deve tenere traccia dell'assegnazione dei buffer ai processi utente
- sottoutilizzo del buffer nel caso di I/O orientato a flusso (problema minore nel caso di gestione di sequenza di caratteri a linee)

Confronto tra single and no-buffering

Prestazioni

T = tempo per l'input di un blocco

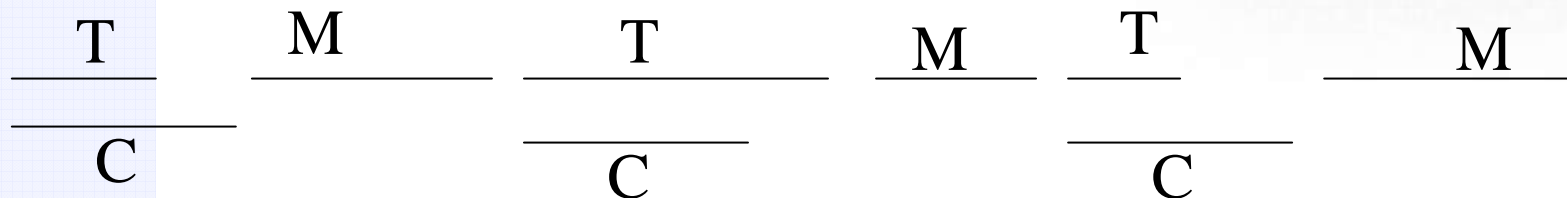
C = tempo di computazione (user process) tra due richieste di input

M = tempo per la copia del blocco nel buffer utente (in genere minore dei tempi precedenti)

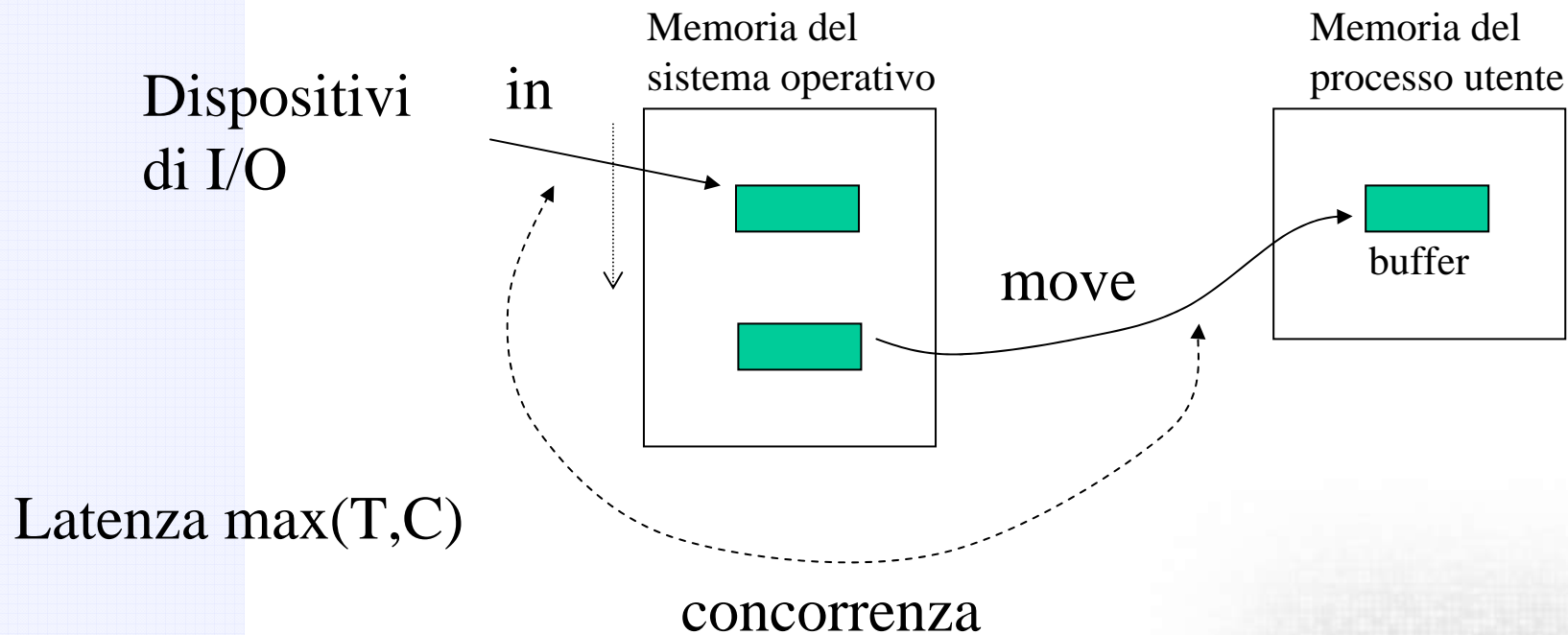
Nel caso "no buffering" tale latenza (o tempo di esecuzione) è T+C



Nel caso "single buffering": Latenza = $\max[C, T] + M$

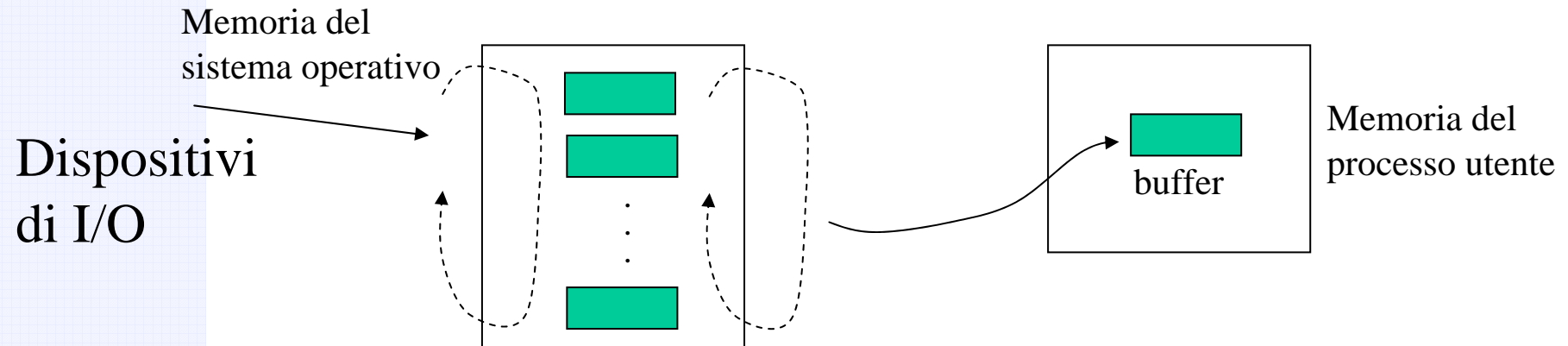


Buffer doppio



- per I/O a blocchi, da vantaggi:
 - se $C < T$ permette di mandare un device I/O a piena velocità
 - se $C > T$ permette ad un processo di non attendere in I/O
- non produce particolari vantaggi nel caso di I/O orientato a flusso di byte

Buffer circolare



- per I/O a blocchi, produce vantaggi nel caso di lunghe sequenze di richieste (burst) di I/O quando $C < T$

Note

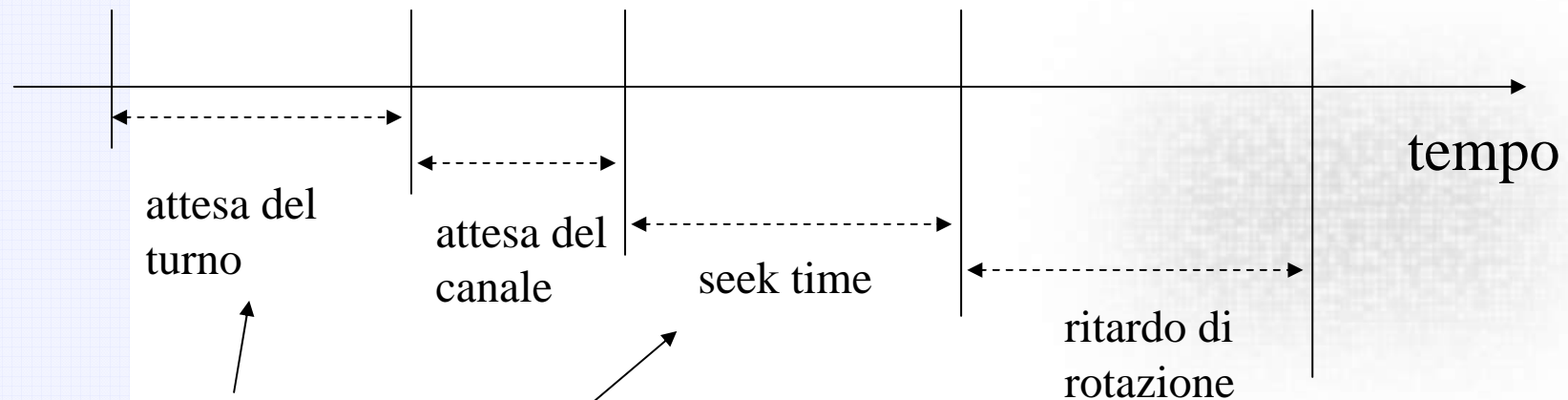
- nessuna quantità di buffer sarà mai sufficiente per evitare il blocco dei processi nel caso $C < T$ ed i processi siano esclusivamente I/O bound
- in pratica i processi esibiscono comportamenti misti (I/O e CPU bound),

Schedulazione del disco

Parametri

- tempo di ricerca della traccia (seek time)
- ritardo di rotazione per l'accesso a settore sulla traccia

Punto di vista del processo



dipendenza dalla
politica di schedulazione

Valutazione dei parametri

Seek time

- n = tracce da attraversare
- m = costante dipendente dal dispositivo
- s = tempo di avvio

$$\Rightarrow T_{seek} = m \times n + s$$

Ritardo di rotazione

- b = bytes da trasferire
- N = numero di bytes per traccia
- r = velocità di rotazione (rev. per min.)

$$\Rightarrow T_{rotazione} = \frac{b}{r \times N}$$

Valori classici

- $s = 20/3$ msec.
- $m = 0.3/0.1$ msec.
- $r = 300/3600$ rpm

Scheduling FCFS

- le richieste di I/O vengono servite nell'ordine di arrivo
- non produce starvation
- non minimizza il seek time

Un esempio

Traccia iniziale = 100

Sequenza delle tracce accedute = 55 - 58 - 39 - 18 - 90 - 160 - 150 - 38 - 184

distanze 45 - 3 - 19 - 21 - 72 - 70 - 10 - 112 - 146

lunghezza media di ricerca

$$\frac{\sum_{i=1}^{|insieme\ dist|} dist_i}{|insieme\ dist|} = 55.3$$

Scheduling su base priorità

- la sequenzializzazione delle richieste di I/O è fuori dal controllo del software di gestione del disco
- si tende a favorire processi con più alta priorità
- l'obiettivo non è l'ottimizzazione dell'utilizzo del disco (non viene minimizzato il seek time)
- rischio di starvation

Scheduling LIFO

- le richieste di I/O vengono servite nell'ordine inverso rispetto all'ordine di arrivo
- può produrre starvation
- minimizza il seek time in caso di accessi sequenziali e allocazione contigua (sistemi di basi di dati)

Scheduling SSTF (Shortest Service Time First)

- si dà priorità alla richiesta di I/O che produce il minor movimento della testina del disco
- non minimizza il tempo di attesa medio
- può provocare starvation

Un esempio

Traccia iniziale = 100

Insieme delle tracce coinvolte = 55 - 58 - 39 - 18 - 90 - 160 - 150 - 38 - 184

Riordino in base al seek time = 90 - 58 - 55 - 39 - 38 - 18 - 150 - 160 - 184

distanze 10 - 32 - 3 - 16 - 1 - 20 - 132 - 10 - 24

lunghezza media di ricerca

$$\frac{\sum_{i=1}^{|insieme\ dist|} dist_i}{|insieme\ dist|} = 27.5$$

Scheduling SCAN

- il seek avviene in una data direzione fino al termine delle tracce o fino a che non ci sono più richieste in quella direzione
- sfavorevole all'area attraversata più di recente (ridotto sfruttamento della località)
- la versione C-SCAN utilizza scansione in una sola direzione

Un esempio

Traccia iniziale = 100

Direzione iniziale = crescente

Insieme delle tracce coinvolte = 55 – 58 – 39 – 18 – 90 – 160 – 150 – 38 – 184

Riordino in base alla direzione di seek = 150 – 160 – 184 – 90 – 58 – 55 – 39 – 38 – 18

distanze 50 – 10 – 24 – 94 – 32 – 3 – 16 – 1 – 20

|insieme dist|

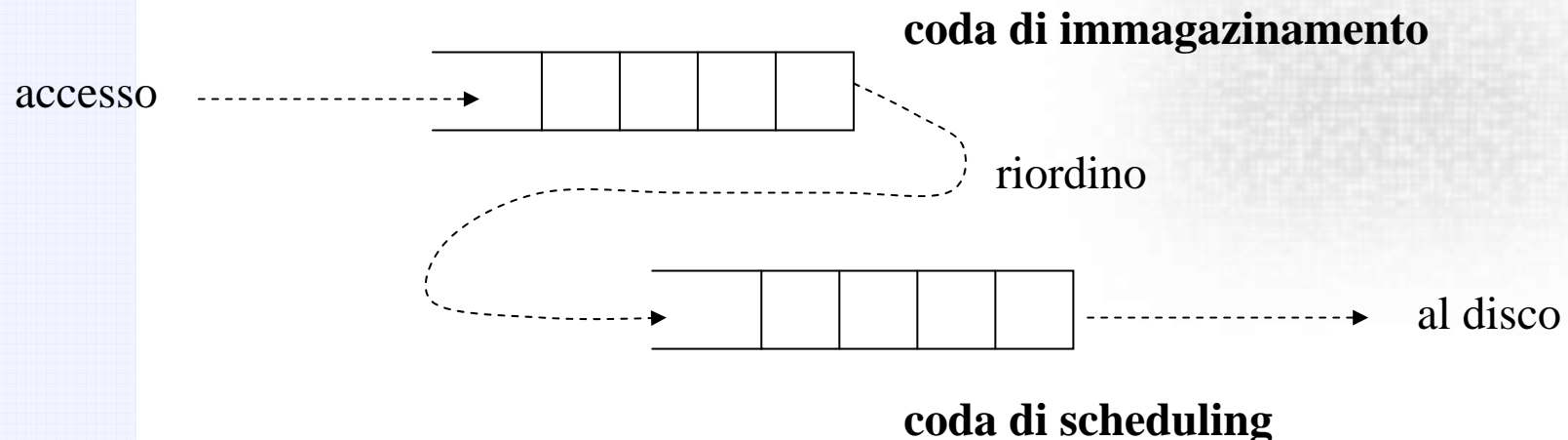
$$\sum_{i=1} dist_i$$

lunghezza media di ricerca

$$\frac{\sum_{i=1} dist_i}{|insieme dist|} = 27.8$$

Scheduling FSCAN

- SSTF, SCAN e C-SCAN possono mantenere la testina bloccata in situazioni patologiche di accesso ad una data traccia
- FSCAN usa due code distinte per la gestione delle richieste
- la schedulazione avviene su una coda
- l'immagazzinamento delle richieste di I/O per la prossima schedulazione avviene sull'altra coda
- nuove richieste non vengono considerate nella sequenza di scheduling già determinata

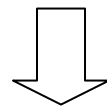


Cache del disco

- il sistema operativo mantiene una regione di memoria che funge da buffer temporaneo (**buffer cache**) per i dati acceduti in I/O
- hit nel buffer cache evita l'interazione con il disco (diminuzione della latenza e del carico su disco)
- efficienza legata alla località delle applicazioni

Strategia di sostituzione dei blocchi

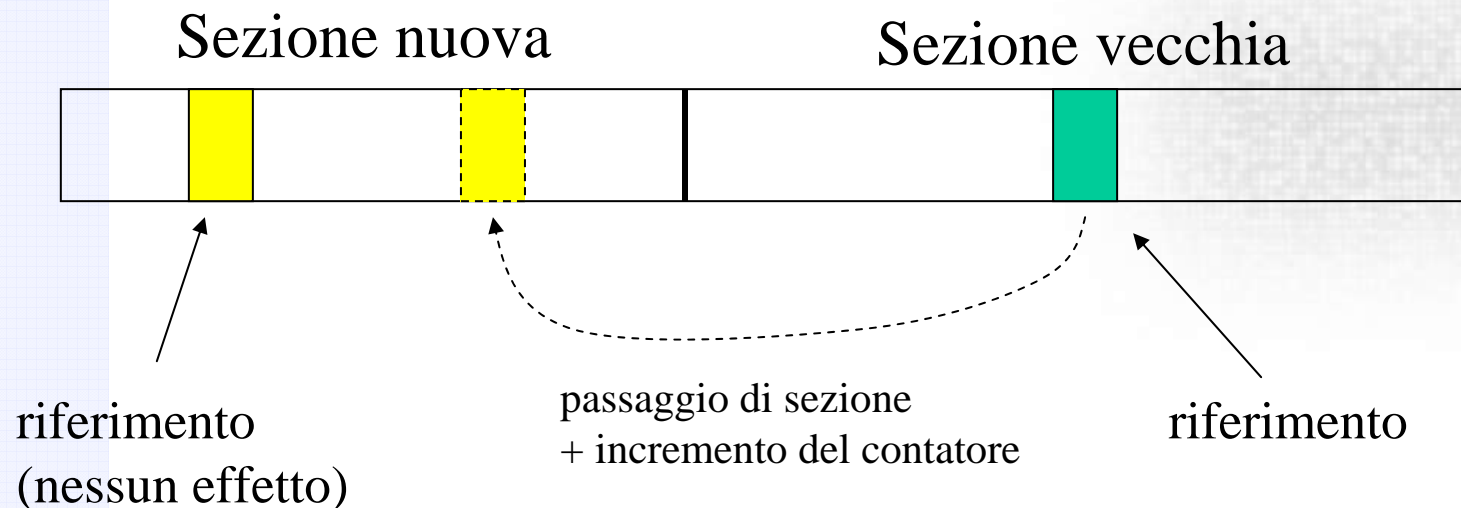
- Least-Recently-Used: viene mantenuta una lista di gestione a stack
- Least-Frequently-Used: si mantiene un contatore di riferimenti al blocco indicante il numero di accessi da quando è stato caricato



Problemi sulla variazione di località

Buffer cache a due sezioni

- ogni volta che un blocco è riferito, il suo contatore di riferimenti è incrementato ed il blocco è portato nella **sezione nuova**
- per i blocchi nella sezione nuova il contatore di riferimenti non viene incrementato
- per la sostituzione dalla sezione nuova si sceglie il blocco con il numero di riferimenti minore
- la stessa politica è usata per la sostituzione nella **sezione vecchia**



Buffer cache a tre sezioni

- esiste una sezione intermedia
- i blocchi della sezione intermedia vengono passati nella sezione vecchia per effetto della politica di sostituzione
- un blocco della sezione nuova difficilmente verrà escluso dal buffer cache in caso sia riferito in breve tempo dall'uscita dalla sezione nuova

