Sistemi Operativi Modulo I

Primo canale (A-L) e Teledidattica
A.A. 2018/2019
Corso di Laurea in Informatica

La Gestione dell'Input/Output

Igor Melatti

Sapienza Università di Roma
Dipartimento di Informatica

SAPIENZA
UNIVERSITÀ DI ROMA

Roadmap

- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

Categorie per i Dispositivi di I/O

- Area assai problematica per la progettazione di sistemi operativi
  - c'è una grande varietà di dispositivi di I/O
  - e anche una grande varietà di applicazioni che li usano
  - difficile scrivere un SO che tenga conto di tutto
- Tre categorie di dispositivi
  - leggibili dall'utente
  - leggibili dalla macchina
  - dispositivi di comunicazione

Dispositivi Leggibili dall'Utente

- Dispositivi usati per la comunicazione diretta con l'utente
  - stampanti
  - "terminali": ovvero monitor + tastiera (+ mouse)
    - da non confondere con i "terminali" software, ovvero applicazioni sulle quali è possibile invocare una shell
    - per "shell" si intende un'applicazione che permette di effettuare syscall in maniera interattiva
    - ad esempio, per esplorare il filesystem
  - joystick
  - ...

Dispositivi Leggibili dalla Macchina

- Dispositivi usati per la comunicazione con materiale elettronico
  - dischi
  - chiavi USB
  - sensori
  - controllori
  - attuatori

Dispositivi per la Comunicazione

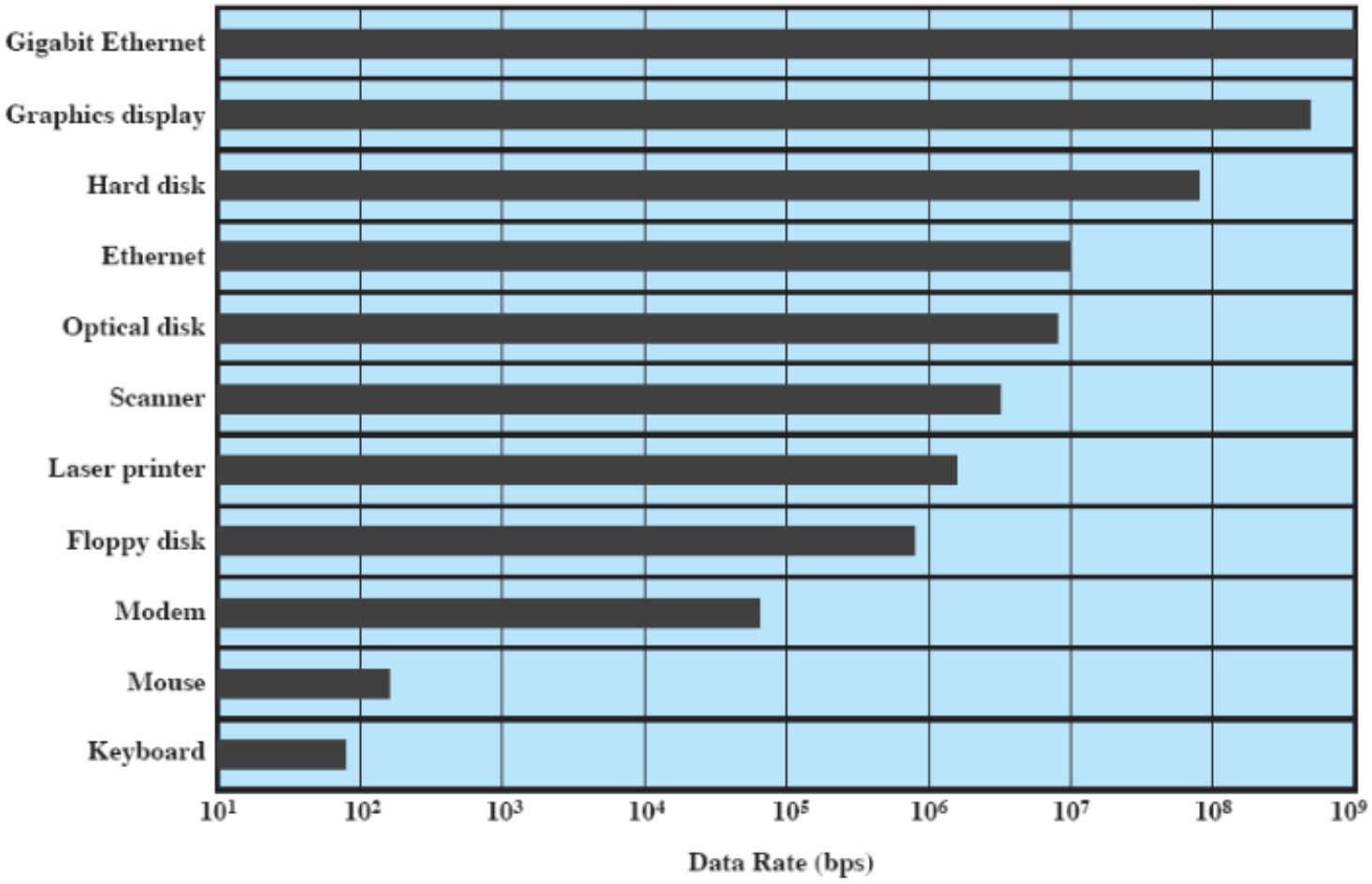
- Dispositivi usati per la comunicazione con dispositivi remoti
  - modem
  - schede Ethernet
  - Wi-Fi

Differenze tra Dispositivi di I/O

- I dispositivi di I/O possono differire sotto molti aspetti
  - data rate (frequenza di accettazione/emissione di dati)
  - applicazione
  - difficoltà nel controllo (tastiera vs. disco...)
  - unità di trasferimento dati (caratteri vs. blocchi)
  - rappresentazione dei dati
  - condizioni di errore

Data Rate

Differenze anche molto elevate



Applicazioni

- I dischi sono usati per memorizzare files, richiedono un software per la gestione dei file
- I dischi sono usati anche per la memoria virtuale, per la quale serve altro software apposito (nonché hardware)
- Un terminale usato da un amministratore di sistema dovrebbe avere una priorità più alta

Complessità del Controllo

- Una tastiera o un mouse richiedono un controllo molto semplice
- Una stampante è più complicata, ma non troppo
- Il disco è tra le cose più complicate da controllare
- Fortunatamente, non si fa tutto col software, e molte cose vengono controllate da hardware dedicato

Unità di Trasferimento Dati

- I dati possono essere trasferiti (dal dispositivo che li genera alla memoria o viceversa):
  - in blocchi di byte di lunghezza fissa
    - usato per la memoria secondaria: dischi, chiavi USB, CD/DVD, ...
  - come un flusso (*stream*) di byte, o equivalentemente di caratteri
    - qualsiasi cosa *non* sia memoria secondaria: terminali, stampanti, schede audio (sia input che output), dispositivi di comunicazione di rete, ...

Rappresentazione dei Dati

- I dati sono rappresentati secondo codifiche diverse in dispositivi diversi
  - una vecchia tastiera magari li rappresenta in ASCII puro, una moderna con l'UNICODE
- Possono essere diversi anche i controlli di parità

Condizioni d'Errore

- La natura degli errori varia di molto da dispositivo a dispositivo
- Ad esempio:
  - nel modo in cui gli errori vengono notificati
  - sulle loro conseguenze (fatali/ignorabili)
  - su come possono essere gestiti

Roadmap

- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

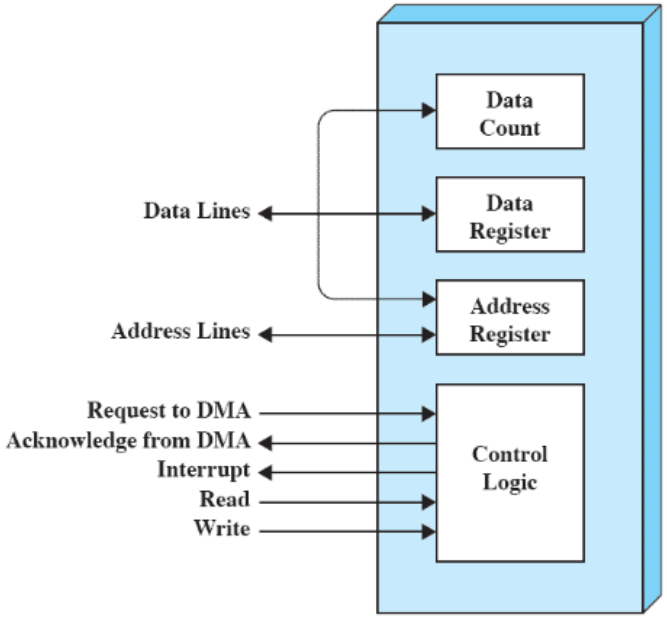
Tecniche per Effettuare l'I/O

- Programmato
- Guidato dagli interrupt
- Accesso diretto in memoria (DMA)

	Senza Interruzioni	**Con Interruzioni**
Passando per la CPU	I/O programmato	I/O guidato dalle interruzioni
Direttam. in memoria		DMA

Direct Memory Access

- Il processore delega le operazioni di I/O al modulo DMA
- Il modulo DMA trasferisce i dati direttamente da o verso la memoria principale
- Quando l'operazione è completata, il modulo DMA genera un interrupt per il processore



Evoluzione della Funzione di I/O

1. Il processore controlla il dispositivo periferico
2. Viene aggiunto un modulo (o controllore) di I/O, direttamente sul dispositivo
   - I/O programmato, senza interrupt
   - ma il processore non si deve occupare di alcuni dettagli del dispositivo stesso
3. Modulo o controllore di I/O con interrupt
   - migliora l'efficienza del processore, che non deve aspettare il completamento dell'operazione di I/O

Evoluzione della Funzione di I/O

4. DMA
   - blocchi di dati viaggiano tra dispositivo e memoria, senza usare il processore
   - il processore fa qualcosa solo all'inizio e alla fine di un'operazione di I/O
5. Il modulo di I/O diventa un processore separato (*I/O channel*)
   - il processore "principale" comanda al processore di I/O di eseguire un certo programma di I/O in memoria principale
6. Processore per l'I/O (*I/O processor* o anche *I/O channel*)
   - ha una sua memoria dedicata
   - usato per le comunicazioni con terminali interattivi

Roadmap

- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

Obiettivo: Efficienza

- La maggior parte dei dispositivi di I/O sono molto lenti rispetto alla memoria principale
- Grazie alla multiprogrammazione, alcuni processi potrebbero essere in attesa del completamento di un'operazione di I/O mentre altri processi sono in esecuzione
- Ma l'I/O potrebbe comunque non tenere il passo con il processore
  - quindi il numero di processi ready si riduce fino a diventare zero
  - si potrebbe pensare che sia sufficiente portare altri processi ready ma sospesi in memoria principale (medium-term scheduler)
  - ma anche questa è un'operazione di I/O
- Necessario quindi cercare soluzioni software dedicate, a livello di SO, per l'I/O
  - in particolare, per il disco

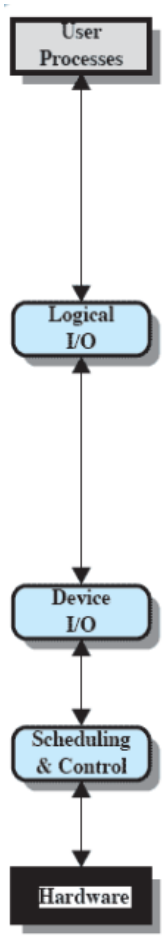
Obiettivo: Generalità

- Per semplicità e per evitare errori, sarebbe bene gestire i dispositivi di I/O, pur nella loro diversità, in modo uniforme
- Nascondere la maggior parte dei dettagli dei dispositivi di I/O nelle procedure di basso livello
- Progettare le funzioni di I/O in modo modulare e gerarchico (anche se è difficile farlo in modo generale)
- Funzionalità da offrire: read, write, lock, unlock, open, close, ...
    - leggere da tastiera o da mouse sono cose molto diverse, ma il SO può farle sembrare uguali o quasi

Progettazione Gerarchica

- Simile al concetto di gerarchia nel progetto di un sistema operativo: insieme di livelli
- Ogni livello si basa sul fatto che il livello sottostante sa effettuare operazioni più primitive
- Ogni livello fornisce servizi al livello superiore
- Ogni livello contiene funzionalità simili per complessità, tempi di esecuzione, livello di astrazione
- Modificare l'implementazione di un livello non dovrebbe avere effetti sugli altri livelli
- Per l'I/O, ci sono 3 macrotipi maggiormente usati

Dispositivo Locale



- Esempi: stampante, monitor, tastiera, ...
- Logical I/O: il dispositivo viene visto come una risorsa logica (open, close, read, ...)
- Device I/O: trasforma richieste logiche in sequenze di comandi di I/O
- Scheduling and Control: esegue e controlla le sequenze di comandi, eventualmente gestendo l'accodamento

Dispositivo di Comunicazione

- Esempi: scheda Ethernet, WiFi, ...
- Come prima, ma al posto del logical I/O c'è una architettura di comunicazione: il dispositivo viene visto come una risorsa logica
- A sua volta, questa consiste in un certo numero di livelli
- Es.: TCP/IP

File System

- Esempi: disco HDD o SDD, scheda di memoria SSD, CD, DVD, floppy disk, USB key ...
- Directory Management: da nomi di file a identificatori di file; tutte le operazioni utente che hanno a che fare con i file (crearli, cancellarli, spostarli, ...)
- File System: struttura logica ed operazioni (apri, chiudi, leggi, scrivi, ...)
- Organizzazione fisica: da identificatori di file a indirizzi fisici su disco; allocazione/deallocazione

Roadmap

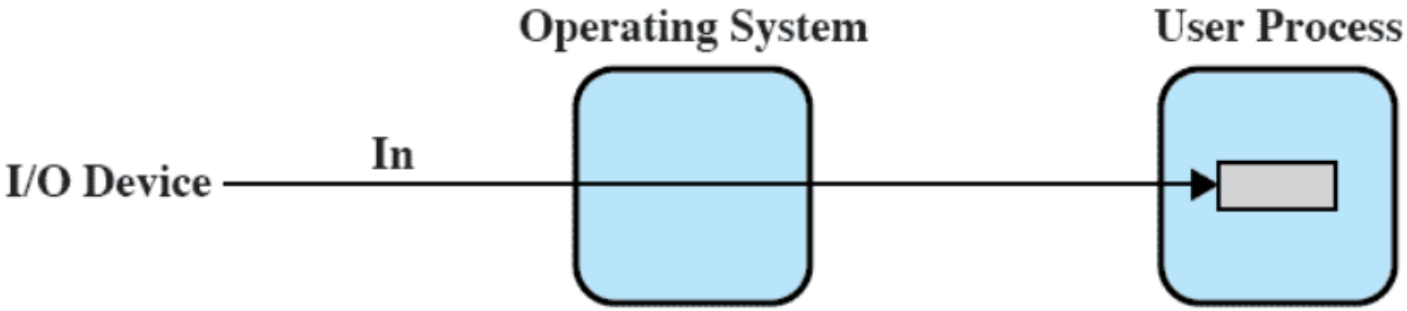
- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

Buffering dell'I/O

- Nell'attesa del completamento di un'operazione di I/O, alcune pagine devono rimanere in memoria principale per evitare il deadlock
  - un processo richiede un I/O su una certa sua zona di memoria (DMA), ma viene subito swappato (quindi sospeso)
  - per poter essere completata, la richiesta di I/O necessita che il processo sia riportato in memoria
  - ma per essere riportato in memoria, la richiesta di I/O dovrebbe essere completata...
- Risolvibile con il frame blocking, ma si limita il numero di pagine swappabili, e le prestazioni del SO decrescono
- Soluzione: *buffering*
  - può essere più efficiente effettuare trasferimenti di input in anticipo e di output in ritardo rispetto alle richieste

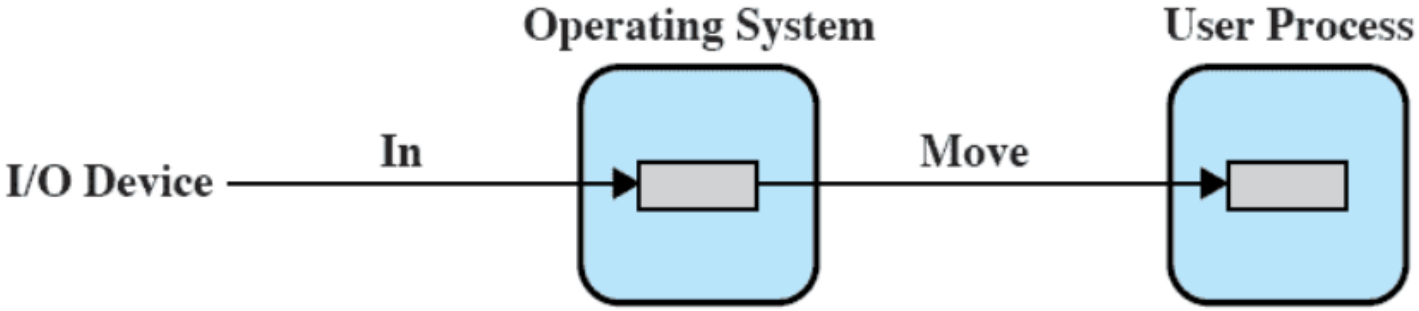
Senza Buffer

Il SO accede al dispositivo nel momento in cui ne ha necessità



Buffer Singolo

Il SO crea un buffer in memoria principale (system space, non user space!) per una data richiesta di I/O



Buffer Singolo Orientato ai Blocchi

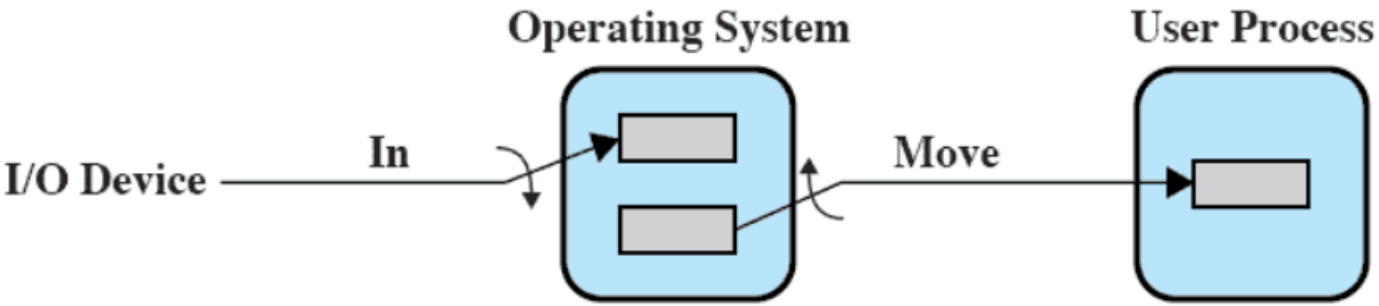
- I trasferimenti di input sono fatti al buffer in system memory
- Blocco viene mandato nello spazio utente quando necessario
- Il prossimo blocco viene comunque letto nel buffer
  - input anticipato (*anticipated input*), o lettura in anticipo (*read ahead*)
  - i dati, solitamente, sono acceduti sequenzialmente: c'è buona probabilità che servirà, e sarà già stato letto
- L'output, invece, viene posticipato
  - per questo serve la system call `flush`...

Buffer Singolo Orientato agli Stream

- I terminali tipicamente hanno a che fare con linee (caratteri più Invio)
- Quindi, si bufferizza una linea intera di input o di output
- Un byte alla volta per i device in cui un singolo carattere premuto va gestito
  - anche per sensori e controllori
  - in pratica, è il produttore/consumatore

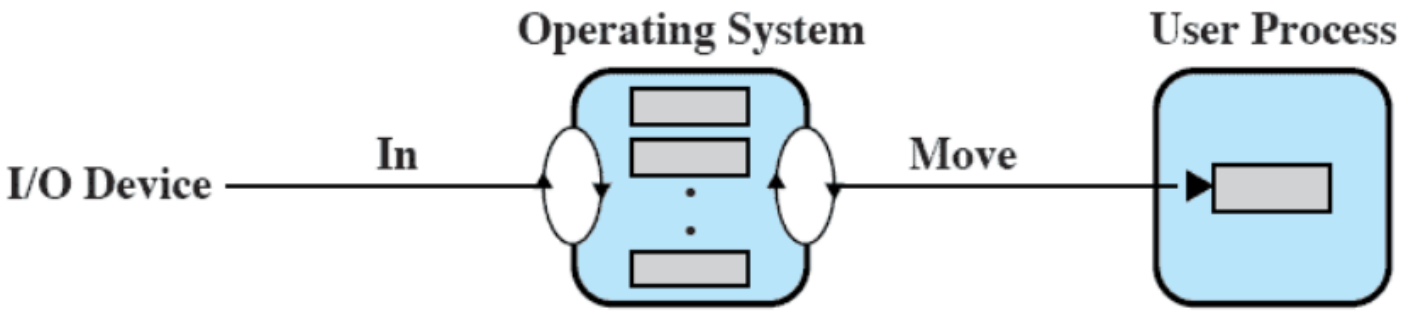
Buffer Doppio

Due buffer anziché uno: un processo può trasferire dati da o a uno dei buffer, mentre il SO svuota o riempie l'altro



Buffer Circolare

- Più di 2 buffer
- Ciascun buffer è un'unità di buffer circolare
- Usato quando l'operazione di I/O deve tenere il passo del processo
- Di nuovo: è proprio il produttore/consumatore!



Buffer: Pro e Contro

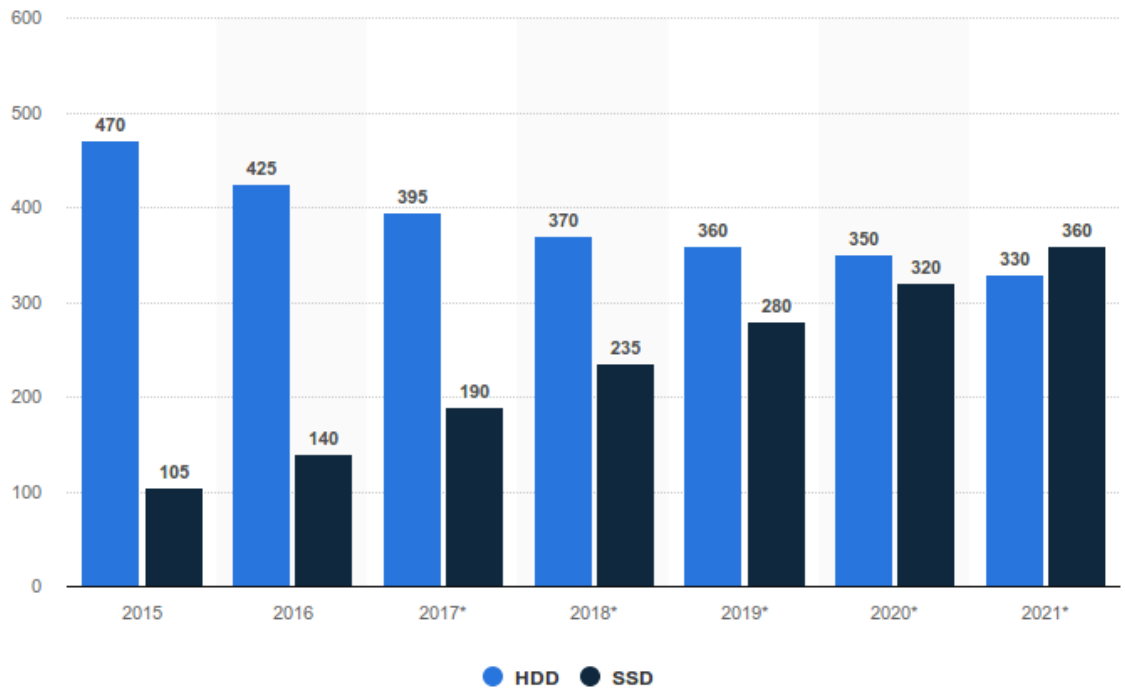
- Il buffering smussa i picchi di richieste di I/O
  - ma se la domanda è molta, i buffer si riempiono e il vantaggio si perde
- Utile soprattutto quando ci sono molti e vari dispositivi di I/O da servire
  - migliora l'efficienza dei processi e del SO

Roadmap

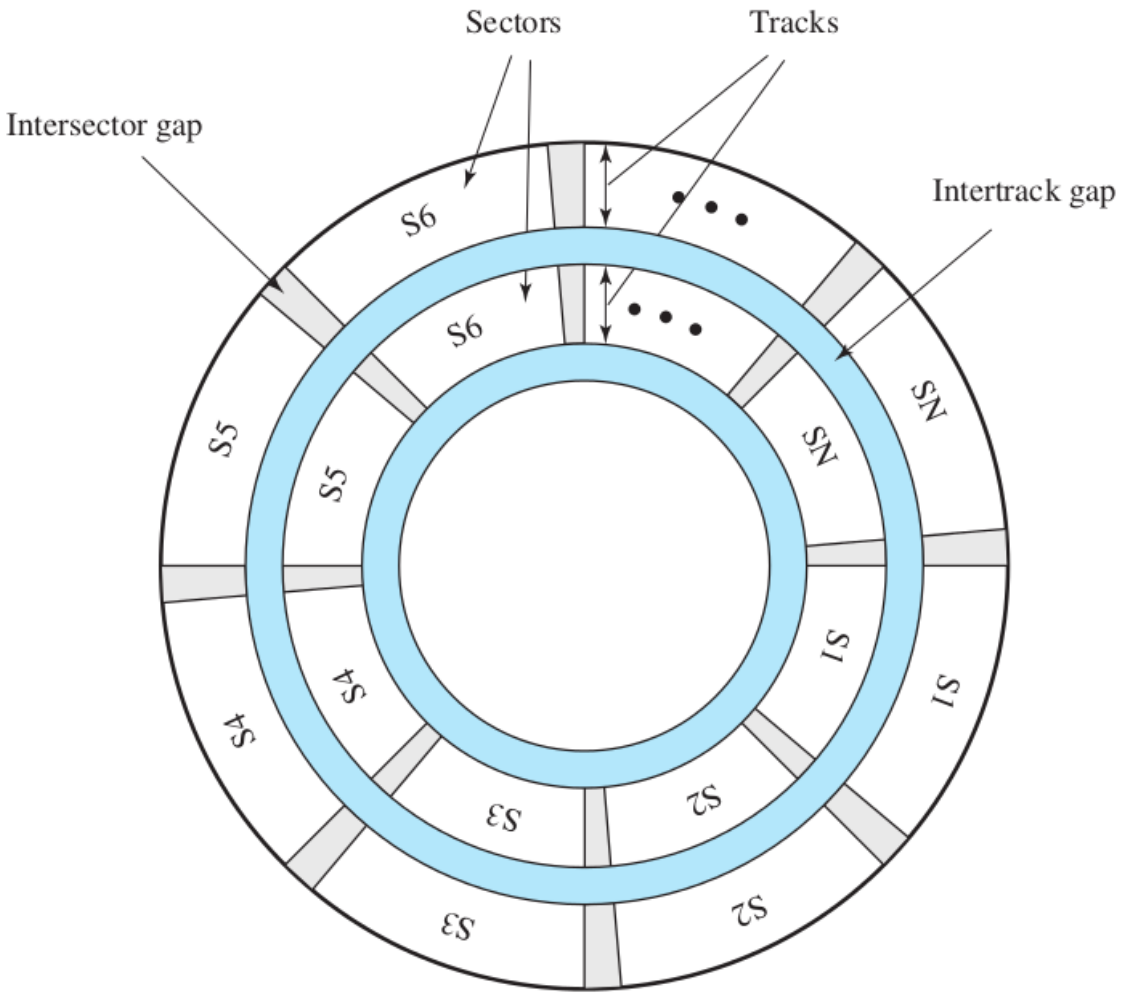
- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

HDD vs SSD

- Quello che diremo vale per gli hard drive disk, non per i solid state disk
- Gli SDD hanno problemi diversi, che noi non tratteremo



Il Disco

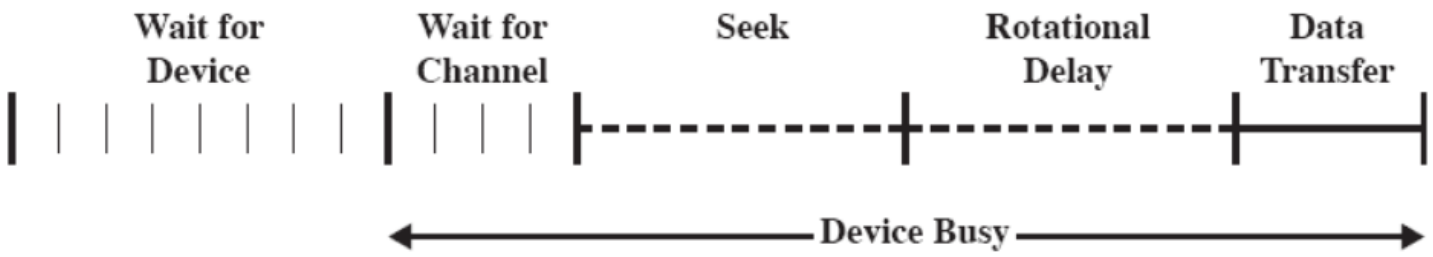

Intersector gap
Sectors
Tracks
Intertrack gap
S1
S2
S3
S4
S5
S6
S1
S2
S3
S4
S5
S6

Come Funziona un Disco

- I dati si trovano sulle tracce (corone concentriche)
  - su un certo insieme di settori
- Quindi per leggere/scrivere occorre sapere su quale traccia si trovano i dati
  - e sulla traccia, su quale settore
- Per selezionare una traccia bisogna:
  - spostare una testina, se il disco ha testine mobili
  - selezionare una testina, se il disco ha testine fisse
- Per selezionare un settore su una traccia bisogna aspettare che il disco ruoti (a velocità costante)
- Se i dati sono tanti, potrebbero essere su più settori o addirittura su più tracce
  - tipica misura di un settore: 512 bytes

Prestazioni del Disco

- Un'operazione su disco dipende da molti dettagli
- La linea temporale generale può essere riassunta come segue



Prestazioni del Disco

Tempo di accesso (*access time*) che è la somma di:

Tempo di posizionamento (*seek time*): tempo necessario perché la testina si posizioni sulla traccia desiderata

Ritardo di rotazione (*rotational delay* o *latency*): tempo necessario affinché l'inizio del settore raggiunga la testina

Tempo di trasferimento (*transfer time*) tempo necessario a trasferire i dati che scorrono sotto la testina

A parte:

- *wait for device*: attesa che il dispositivo sia assegnato alla richiesta
- *wait for channel*: attesa che il sottodispositivo sia assegnato alla richiesta
  - se ci sono più dischi che condividono un unico canale di comunicazione

Politiche di Scheduling per il Disco

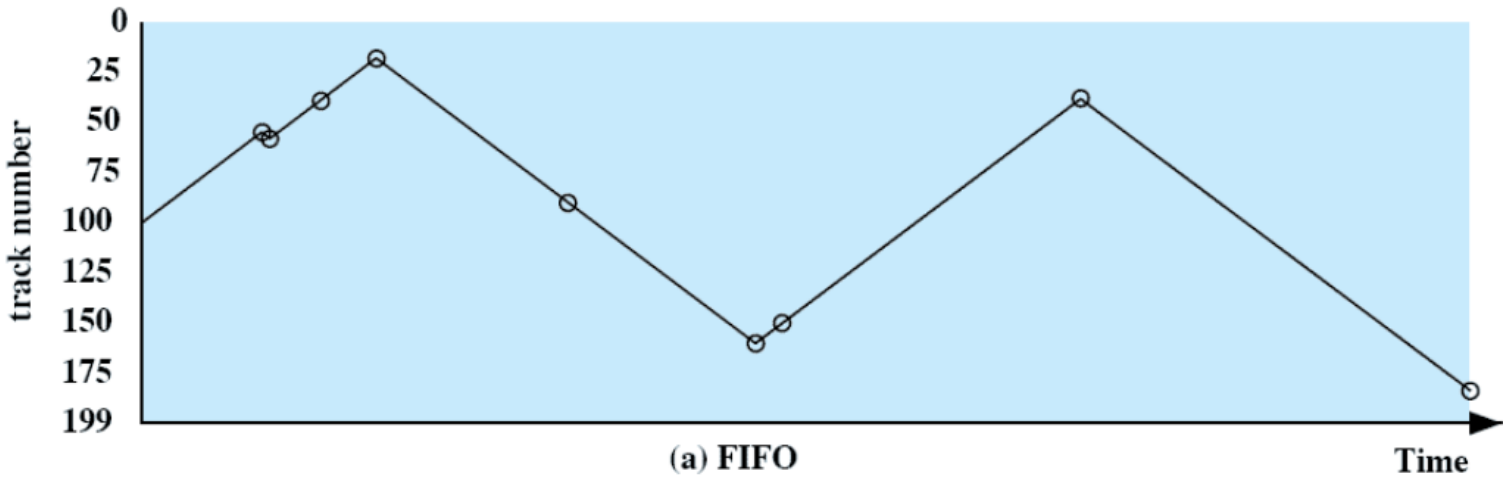
- Le confronteremo su un esempio comune
- All'inizio, la testina si trova sulla traccia numero 100
- Ci sono 200 tracce
- Vengono richieste le seguenti tracce nel seguente ordine:

$$55, 58, 39, 18, 90, 160, 150, 38, 184$$

- Considereremo solo il seek time, che è il parametro più importante per le prestazioni
- Confronto con il random, che è lo scheduling peggiore

FIFO

- Le richieste sono servite in modo sequenziale
- Equo nei confronti dei processi
- Se ci sono molti processi in esecuzione, le prestazioni sono simili allo scheduling random



Priorità

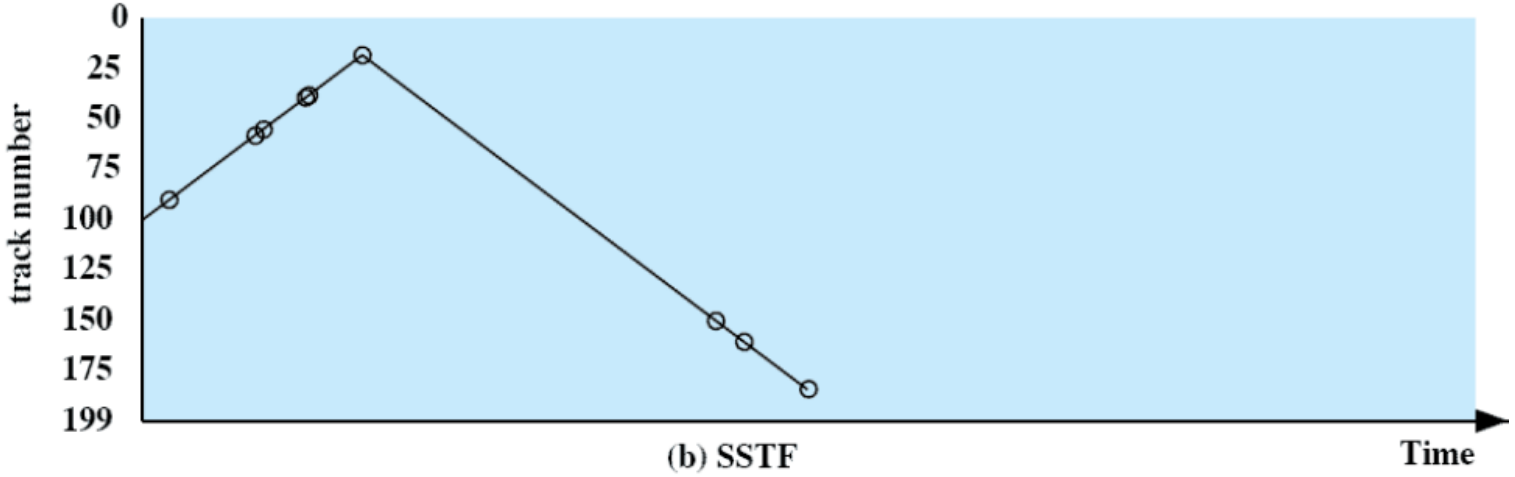
- L'obiettivo non è ottimizzare il disco, ma raggiungere altri obiettivi
- I processi batch corti potrebbero avere priorità più alta
- È desiderabile fornire un buon tempo di risposta ai processi interattivi
- Ma i processi più lunghi potrebbero dover aspettare troppo
- Non va bene per i DBMS
- Impossibile fare il grafico: bisognerebbe sapere, per ogni richiesta, qual è la priorità del processo che l'ha fatta

LIFO

- Ottimo per DBMS con transazioni
- Il dispositivo è dato all'utente più recente, così da minimizzare il movimento del braccio
- Possibile la starvation
- Impossibile fare il grafico: bisognerebbe sapere, per ogni richiesta, a quale utente appartiene

Minimo Tempo di Servizio

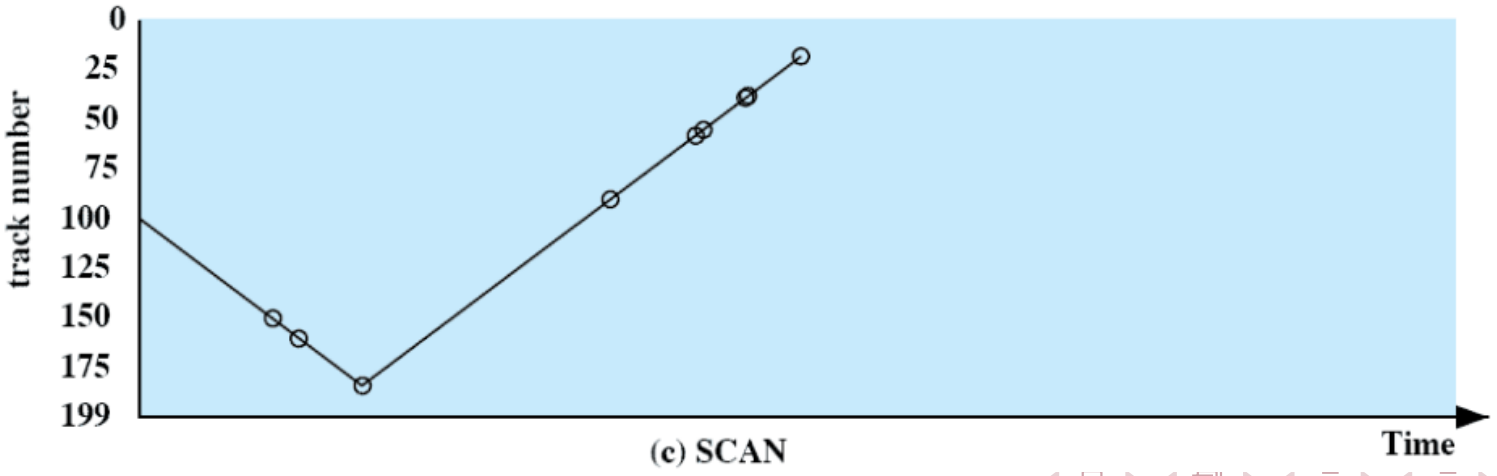
- Da qui in poi, occorre conoscere la posizione della testina
- Sceglie la richiesta che minimizza il movimento del braccio (dalla posizione attuale)
- Quindi sceglie sempre il tempo di posizionamento minore
  - ovviamente, rispetto alle richieste attualmente pervenute
- Possibile la starvation delle richieste: arrivano continuamente richieste più vicine



(b) SSTF

SCAN

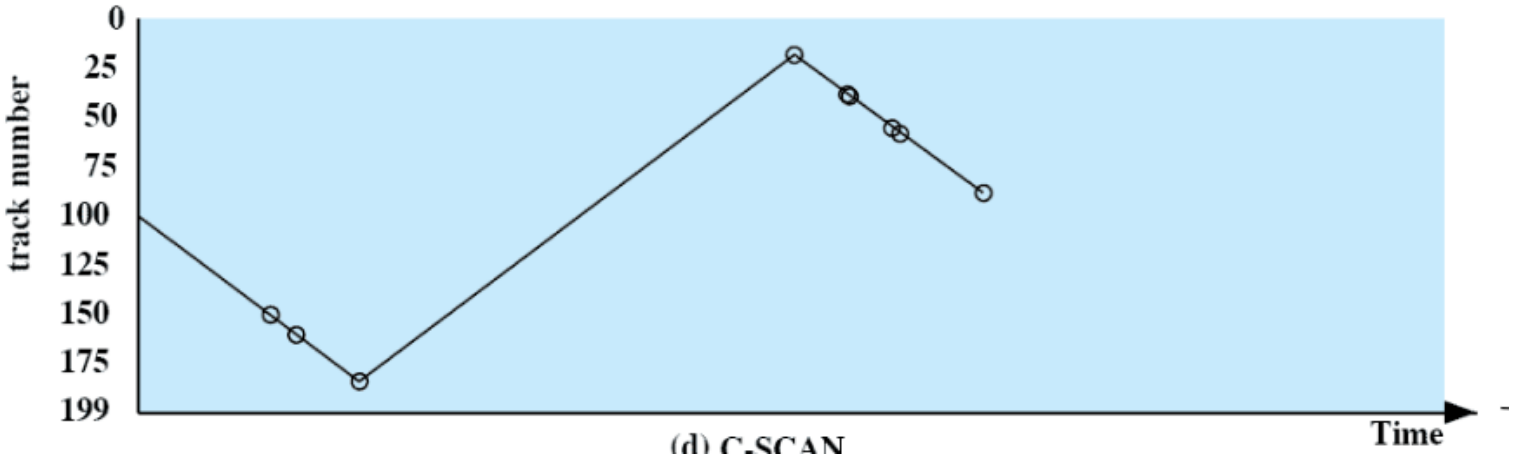
- Si scelgono le richieste in modo tale che il braccio si muova sempre in un verso, e poi torni indietro
- Niente starvation delle richieste, ma è poco "fair":
  - favorisce le richieste ai bordi (attraversati 2 volte in poco tempo...) $\rightarrow$ C-SCAN
  - favorisce le richieste appena arrivate $\rightarrow$ N-step-SCAN
    - nel senso che, se ben piazzate rispetto alla testina, possono precedere richieste che attendono da molto



(c) SCAN

C-SCAN

- Come SCAN, ma niente marcia indietro
- O meglio, nella marcia indietro non si scelgono richieste
- Quindi i bordi non sono più visitati 2 volte in poco tempo



FSCAN

- Due code anziché una
- Quando SCAN inizia, tutte le richieste sono nella coda F, e l'altra coda R è vuota
- Mentre SCAN serve tutta F, ogni nuova richiesta è aggiunta ad R
- Quando SCAN finisce di servire F, si scambiano F ed R
- Ogni nuova richiesta deve aspettare che tutte le precedenti vengano servite
- Quindi le richieste vecchie non sono sopravanzate come in SCAN

N-step-SCAN

- Generalizzazione di FSCAN a $N > 2$ code
- Si accodano le richieste nella coda i-esima finché non si riempie; poi si passa alla $(i + 1) \bmod N$
- Non sono mai aggiunte a quella attualmente servita
- Se N è alto, le prestazioni sono quelle di SCAN (ma con fairness)
- Se $N = 1$, allora si usa il FIFO per evitare di essere unfair

Confronto Prestazionale

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Prospetto

Algoritmo	Descrizione	Note
Selezione a cura del richiedente		
RSS	Scheduling random	Solo per simulazioni e confronti
FIFO	A coda semplice	Il più equo
PRI	Priorità del processo	Non è il disco che decide
Selezione sulla base del dato richiesto		
SSTF	Tempo di servizio più corto	Alto uso del disco, piccole code
SCAN	Avanti ed indietro sul disco	Miglior distribuzione del servizio
C-SCAN	Avanti, con ritorno veloce	Minore variabilità di servizio
N-step-SCAN	SCAN su N richieste	Garanzia del servizio
FSCAN	N-step-SCAN con due code	Tiene conto del carico

Roadmap

- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

Cache del Disco

- È un buffer in memoria principale, usato per i settori del disco
- Contiene una copia di alcuni settori del disco
- Quando si fa una richiesta di I/O per dati che si trovano su un certo settore, si vede prima se tale settore è nella cache
- Se non c'è, il settore letto viene anche copiato nella cache
- Ci sono svariati metodi per gestire questa cache
- Spesso chiamata *page cache*; da non confondere con la cache spesso presente direttamente sui dischi
  - quest'ultima è hardware e quindi trasparente per il sistema operativo
  - `https://en.wikipedia.org/wiki/Page_cache`

Usato Meno di Recente (LRU)

- Se occorre rimpiazzare qualche settore nella cache piena, si prende quello nella cache da più tempo senza referenze
- La cache viene puntata da uno "stack" di puntatori
  - quello riferito più di frequente è alla cima dello stack
  - quindi, ogni volta che un settore viene referenziato o copiato nella cache, il suo puntatore viene portato in cima allo stack
  - non è un vero stack, perché non è acceduto usando solo push, pop e top

Usato Meno di Frequente (LFU)

- Si rimpiazza il settore con meno referenze
- Ovviamente, serve un contatore per ogni settore
  - incrementato ad ogni riferimento
  - inizialmente 1
  - si sceglie il settore col contatore minimo
- Sembra più sensato di LRU: meno vieni usato, meno servi
- Ma la località potrebbe avere un effetto curioso:
  - un settore viene acceduto svariate volte di fila, perché contiene dati acceduti secondo il principio di località
  - dopodiché, non serve più
  - a questo punto, ha un valore abbastanza alto, ma andrebbe invece sostituito

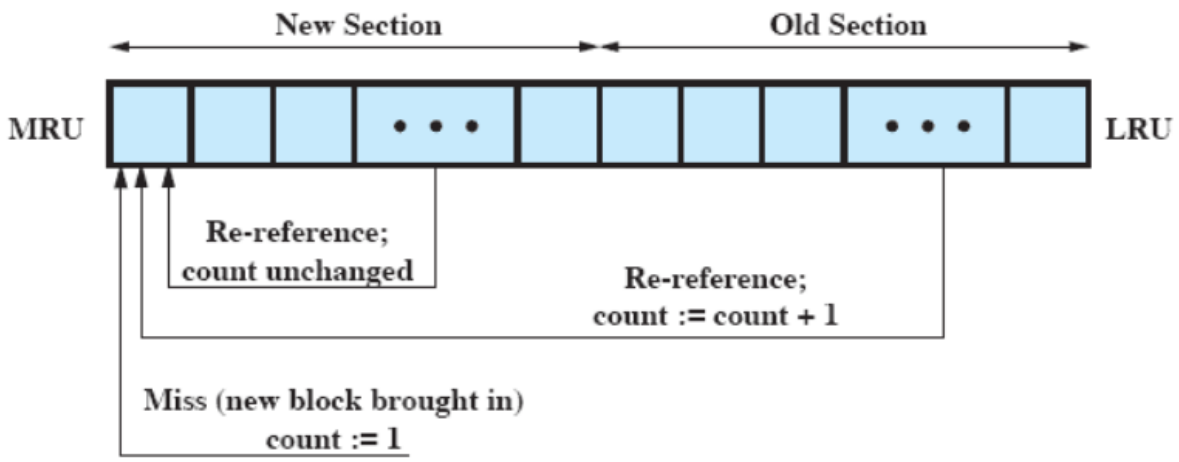
Sostituzione Basata su Frequenza

- C'è uno "stack" di puntatori, come nell'LRU
  - quando un blocco viene referenziato, lo si sposta all'inizio dello stack
- Ma è spezzato in 2: una parte nuova e una vecchia
  - come LFU, si incrementa un contatore ad ogni riferimento, ma solo se si è nella parte vecchia
  - e si sceglie il blocco con contatore minimo *che sia nella parte vecchia*
    - in caso di parità, quello più recente
  - si passa dalla parte nuova alla vecchia per scorrimento: quando un blocco vecchio viene riferito e diventa nuovo, spinge l'ultimo dei nuovi a diventare primo dei vecchi
- Ancora non ci siamo:
  - se non c'è *presto* un riferimento ad un blocco, questo potrebbe essere sostituito solo perché è appena arrivato nella parte vecchia
  - ma magari serviva, i riferimenti sarebbero arrivati di lì a poco

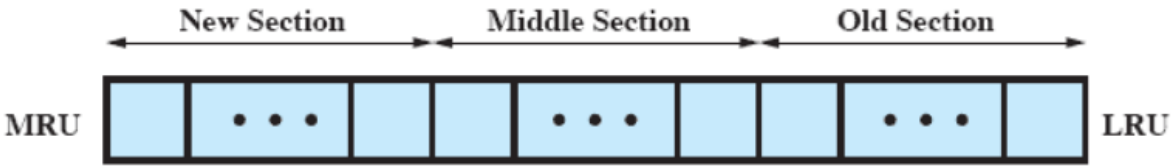
Sostituzione Basata su Frequenza: 3 Segmenti

- Nuovo
  - unica parte dove i contatori non vengono incrementati
  - non eleggibile per rimpiazzamento
- Medio
  - i contatori vengono incrementati, ma ancora non eleggibile per il rimpiazzamento
- Vecchio
  - i contatori vengono incrementati e i blocchi sono eleggibili per il rimpiazzamento
- Risolve i problemi visti sopra

Sostituzione Basata su Frequenza



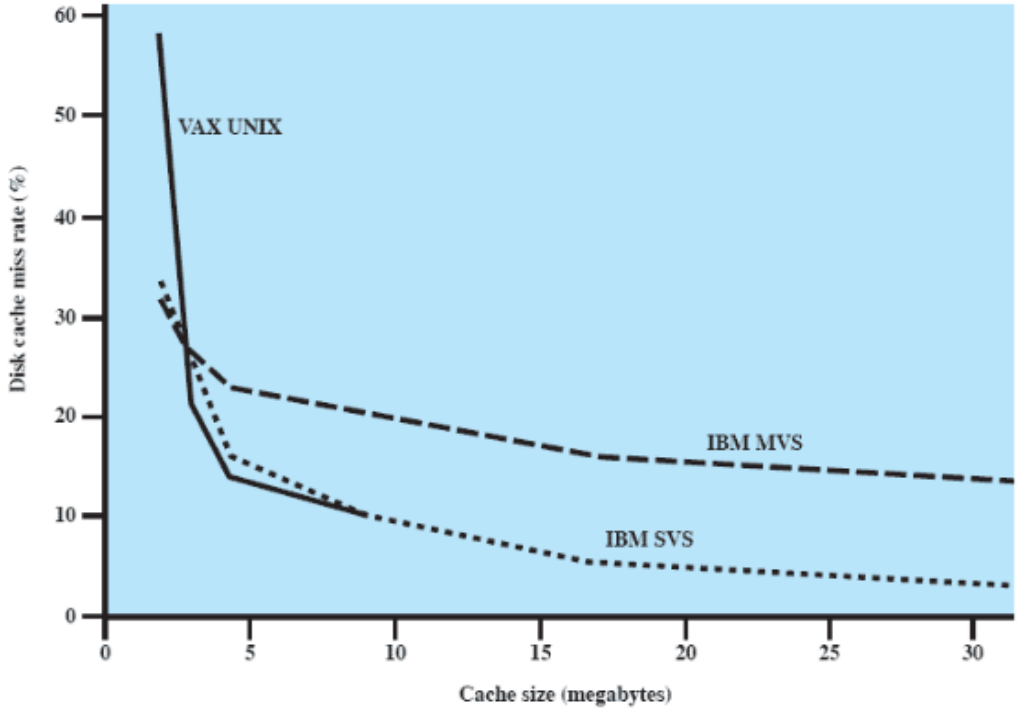
(a) FIFO



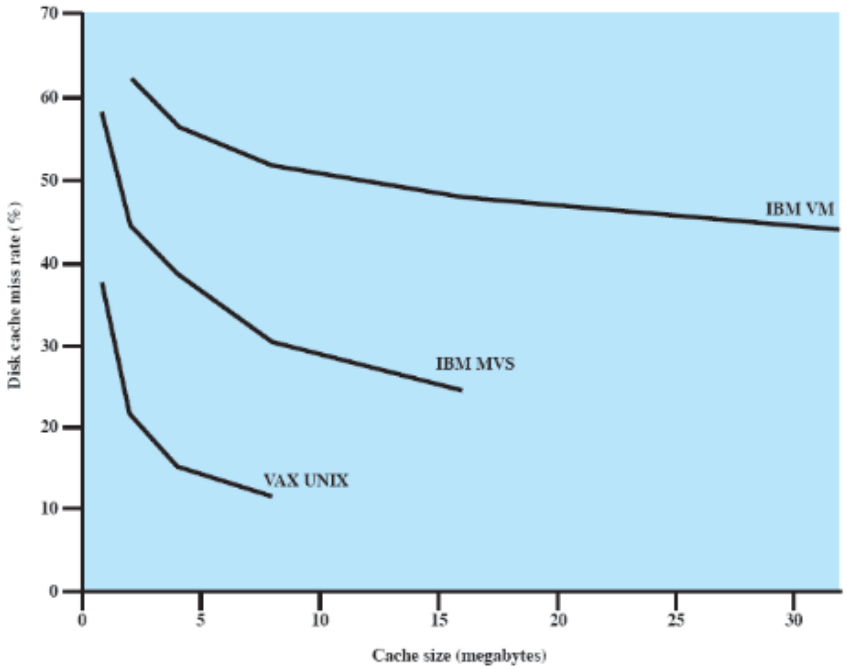
(b) Use of three sections

Prestazioni della Cache del Disco

LRU



LFU



LRU meglio perché sono studi diversi: se la sequenza è la stessa, viene meglio LFU

Roadmap

- Dispositivi di I/O
- Organizzazione della funzione di I/O
- Problemi nel progetto del sistema operativo
- Buffering dell'I/O
- Scheduling del disco
- Cache del disco
- I/O in Linux

Page Cache in Linux

- Unica page cache per tutti i trasferimenti tra disco e memoria
  - compresi quelli dovuti alla gestione della memoria virtuale
- 2 vantaggi:
  - scritture condensate
  - sfruttando la località dei riferimenti, si risparmiano accessi a disco
- Scrittura su disco quando:
  - è rimasta poca memoria: una parte della page cache è ridestinata ad uso diretto dei processi
  - quando l'età delle pagine "sporche" va sopra una certa soglia
- Niente politica separata di replacement: è la stessa usata per il rimpiazzamento delle pagine
  - ovvero: la page cache è paginata, e le sue pagine sono rimpiazzate con l'algoritmo visto per la gestione della memoria