Capitolo 12

Memoria secondaria e terziaria

OBIETTIVI

- **Descrizione della struttura fisica dei dispositivi di memorizzazione secondaria e terziaria e degli effetti che derivano dal loro utilizzo.**
- **Spiegazione delle caratteristiche di prestazione della memoria secondaria e terziaria.**
- **Analisi dei servizi offerti dal sistema operativo per la memorizzazione secondaria e terziaria, compresi** RAID **e** HSM**.**

Il file system, da un punto di vista logico, si può considerare composto da tre parti: nel Capitolo 10 è stata presentata l'interfaccia per il programmatore e per l'utente del file system; nel Capitolo 11 sono state descritte le strutture dati interne e gli algoritmi usati dal sistema operativo per realizzare quest'interfaccia; nel presente capitolo si analizza il livello più basso del file system, e cioè la struttura dei supporti di memorizzazione secondaria e terziaria. Si descrivono innanzitutto gli algoritmi di scheduling delle unità a disco, che ordinano la sequenza delle operazioni di I/O al fine di migliorare le prestazioni del sistema. Quindi si illustrano la formattazione dei dischi e la gestione dei blocchi d'avviamento, dei blocchi danneggiati e dell'area d'avvicendamento dei processi. Si esamina la struttura della memoria secondaria, trattando l'affidabilità dei dischi e la realizzazione della memoria stabile. Il capitolo termina con una breve descrizione dei dispositivi di memoria terziaria e dei problemi che si presentano con il loro utilizzo da parte del sistema operativo.

12.1 Struttura dei dispositivi di memorizzazione

In questo paragrafo si introduce una presentazione generale della struttura fisica dei dispositivi di memorizzazione secondaria e terziaria.

12.1.1 Dischi magnetici

I **dischi magnetici** sono il mezzo fondamentale di memoria secondaria dei moderni sistemi di calcolo. Concettualmente, i dischi (Figura 12.1) sono relativamente semplici: i **piatti** dei dischi hanno una forma piana e rotonda come quella dei CD, con un diametro che comunemente varia tra 1,8 e 5,25 pollici, e le due superfici ricoperte di materiale magnetico; le informazioni si memorizzano registrandole magneticamente sui piatti.

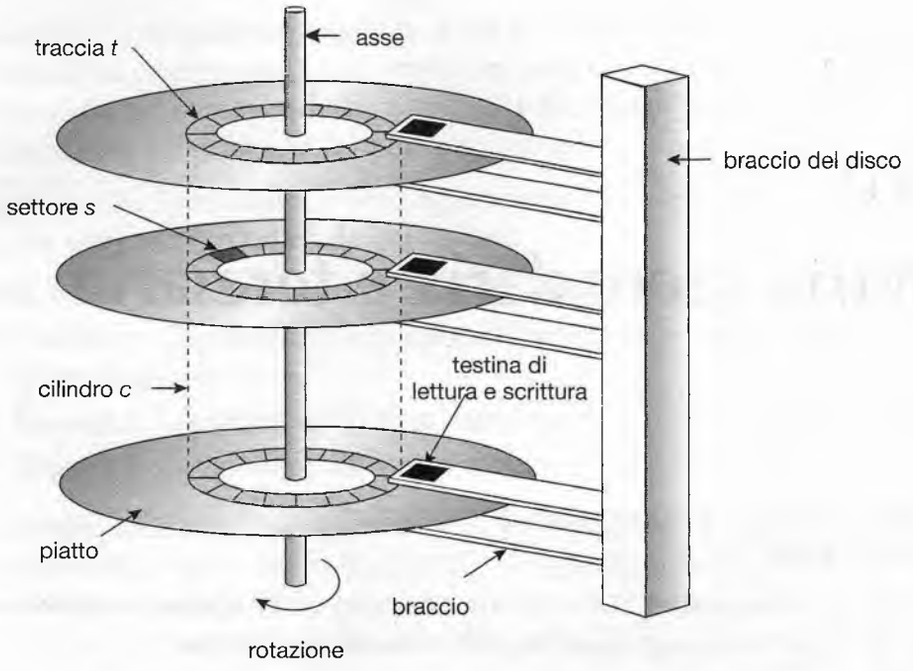


Figura 12.1 Schema funzionale di un disco.

Le testine di lettura e scrittura sono sospese su ciascuna superficie d'ogni piatto e sono attaccate al **braccio del disco** che le muove in blocco. La superficie di un piatto è divisa logicamente in **tracce** circolari a loro volta suddivise in **settori**; l'insieme delle tracce corrispondenti a una posizione del braccio (equidistanti dal centro dei piatti) costituisce un **cilindro**. In un'unità a disco possono esservi migliaia di cilindri concentrici e ogni traccia può contenere centinaia di settori. La capacità di memorizzazione di una comune unità a disco è dell'ordine delle decine di gigabyte.

Quando un disco è in funzione, un motore lo fa ruotare ad alta velocità; la maggior parte dei dischi ruota a velocità comprese tra 60 e 200 giri al secondo. L'efficienza di un disco è caratterizzata da due valori: la **velocità di trasferimento**, cioè la velocità con cui i dati fluiscono dall'unità a disco al calcolatore, e il **tempo di posizionamento**, talvolta detto tempo d'accesso diretto, che consiste nel tempo necessario a spostare il braccio del disco in corrispondenza del cilindro desiderato, detto **tempo di ricerca** (*seek time*), e nel tempo necessario affinché il settore desiderato si porti, tramite la rotazione del disco, sotto la testina, detto **latenza di rotazione**. In genere i dischi possono trasferire parecchi megabyte di dati al secondo e hanno un tempo di ricerca e una latenza di rotazione di diversi millisecondi.

Poiché le testine di un disco sono sospese su un cuscino d'aria sottilissimo (dell'ordine dei micron), esiste il pericolo che la testina urti la superficie del disco; in tal caso, nonostante i piatti del disco siano ricoperti da un sottile strato protettivo, la testina può danneggiare la superficie magnetica. Tale incidente, detto **crollo della testina**, di solito non può essere riparato e comporta la sostituzione dell'intera unità a disco.

Un disco può essere **rimovibile**: ciò permette che diversi dischi siano montati secondo le necessità. I dischi magnetici rimovibili consistono generalmente in un piatto contenuto in un involucro di materiale plastico, che serve a evitare danni che possono verificarsi quando il disco non è inserito nella propria unità. I dischetti (*floppy disk*) sono dischi magnetici economici e rimovibili costituiti da un involucro di plastica che contiene un piatto flessibile. Poiché generalmente la testina delle unità a dischetti poggia direttamente sulla superficie

del disco, per ridurne l'usura, si fa ruotare il dischetto molto più lentamente dei piatti di un'unità a disco (*hard disk*). La capacità di memorizzazione di un dischetto è dell'ordine di circa 1,44 MB. Sono disponibili dischi rimovibili che funzionano in modo assai simile alle normali unità a disco e che hanno capacità dell'ordine dei gigabyte.

Un'unità a disco è connessa a un calcolatore attraverso un insieme di fili detto **bus di I/O**; esistono diversi tipi di tale bus, tra i quali i bus EIDE (*enhanced integrated drive electronics*), ATA (*advanced technology attachment*), ATA **seriale** (*serial ATA*), USB (*universal serial bus*), FC (*fiber channel*) e SCSI. Il trasferimento dei dati in un bus è eseguito da speciali unità di elaborazione dette **controllori**: gli **adattatori** (*adapter*) o **controllori di macchina** (*host controller*) sono i controllori posti all'estremità relativa al calcolatore del bus; i **controllori dei dischi** (*disk controller*) sono incorporati in ciascuna unità a disco. Per eseguire un'operazione di I/O il calcolatore inserisce un comando nell'adattatore, generalmente mediante porte di I/O mappate in memoria, com'è descritto nel Paragrafo 9.7.3; l'adattatore invia il comando al controllore del disco, che agisce sugli elementi elettromeccanici dell'unità a disco per portare a termine il comando. I controllori dei dischi di solito hanno una cache incorporata: il trasferimento dei dati nell'unità a disco avviene tra la cache e la superficie del disco; il trasferimento dei dati tra la cache e l'adattatore avviene alla velocità propria dei dispositivi elettronici.

12.1.2 Nastri magnetici

I **nastri magnetici** sono stati i primi supporti di memorizzazione secondaria. Pur avendo la capacità di memorizzare in modo permanente un'enorme quantità di dati, queste unità sono caratterizzate da un tempo d'accesso molto elevato rispetto a quello della memoria centrale e dei dischi magnetici. Inoltre il tempo d'accesso diretto dei nastri magnetici (essendo fisicamente ad accesso sequenziale) è migliaia di volte maggiore di quello dei dischi magnetici, e ciò li rende inadatti come supporto di memoria secondaria. Gli usi principali dei nastri sono la creazione di copie di riserva dei dati (*backup*), la registrazione di dati poco usati e il trasferimento di informazioni tra diversi sistemi di calcolo.

Il nastro è avvolto in bobine e scorre su una testina di lettura e scrittura. Il posizionamento sul settore richiesto può richiedere alcuni minuti, anche se, una volta raggiunta la posizione desiderata, l'unità a nastro può leggere o scrivere informazioni a una velocità paragonabile a quella di un'unità a disco. La capacità varia secondo il particolare tipo di unità a nastro. I nastri hanno in genere una capacità compresa fra 20 e 200 GB. Alcune unità sono dotate di funzionalità di compressione dei dati che permettono di raddoppiare la capacità effettiva. Le unità a nastro e i loro driver sono solitamente classificate per larghezza: misure tipiche sono 4, 8 o 19 millimetri, e 1/4 o 1/2 pollice. Alcune unità prendono il nome dalla tecnologia su cui si fondano, come nel caso di LTO-2 e SDLT. Per altre informazioni sui nastri si veda il Paragrafo 12.9.

VELOCITÀ DI TRASFERIMENTO DEL DISCO

Come per molti altri aspetti dell'informatica, le prestazioni pubblicate relative ai dischi non coincidono con le cifre reali: sono sempre inferiori delle **velocità di trasferimento effettive**, per esempio. La velocità di trasferimento può essere considerata la velocità con cui la testina legge i bit dal supporto magnetico, ma questo aspetto va distinto dalla velocità con cui i blocchi sono consegnati al sistema operativo.

FIREWIRE

FireWire è il nome di un'interfaccia progettata per collegare a un computer periferiche diverse, quali DVD, videocamere digitali e dischi rigidi. Si tratta di un prodotto originariamente sviluppato da Apple, e poi recepito come standard IEEE 1394 nel 1995. Nella sua prima versione, lo standard prevedeva una larghezza di banda massima di 440 megabit al secondo. Recentemente è apparso lo standard IEEE 1394b per FireWire2, che prevede fino a 800 megabit al secondo di larghezza di banda, circa il doppio dell'originale.

12.2 Struttura dei dischi

I moderni dischi sono considerati come grandi array monodimensionali di **blocchi logici**, dove un blocco logico è la minima unità di trasferimento. La dimensione di un blocco logico è di solito di 512 byte, sebbene alcuni dischi si possano **formattare a basso livello** allo scopo di ottenere una diversa dimensione dei blocchi logici, ad esempio 1024 byte; per altre informazioni su quest'opzione, si veda il Paragrafo 12.5.1. L'array monodimensionale di blocchi logici corrisponde in modo sequenziale ai settori del disco: il settore 0 è il primo settore della prima traccia sul cilindro più esterno; la corrispondenza prosegue ordinatamente lungo la prima traccia, quindi lungo le rimanenti tracce del primo cilindro, e così via di cilindro in cilindro dall'esterno verso l'interno.

Sfruttando questa corrispondenza sarebbe possibile – almeno in teoria – trasformare il numero di un blocco logico in un indirizzo fisico di vecchio tipo, consistente in un numero di cilindro, un numero di traccia concernente quel cilindro, e un numero di settore relativo a quella traccia. In pratica, però, vi sono due motivi che rendono difficile quest'operazione: in primo luogo, la maggior parte dei dischi contiene settori difettosi, ma la corrispondenza nasconde questo fatto sostituendo ai settori malfunzionanti settori sparsi in altre parti del disco; in secondo luogo, il numero di settori per traccia in certe unità a disco non è costante.

Nei supporti che impiegano la **velocità lineare costante** (*constant linear velocity*, CLV) la densità di bit per traccia è uniforme. Più è lontana dal centro del disco, tanto maggiore è la lunghezza della traccia, tanto maggiore è il numero di settori che essa può contenere. Spostandosi da aree esterne verso aree più interne il numero di settori per traccia diminuisce. Le tracce nell'area più esterna contengono in genere il 40 per cento in più dei settori contenuti nelle tracce dell'area più interna. L'unità aumenta la sua velocità di rotazione man mano che le testine si spostano dalle tracce esterne verso le tracce più interne per mantenere costante la quantità di dati che scorrono sotto le testine. Questo metodo si usa nelle unità per CD-ROM e DVD. In alternativa la velocità di rotazione dei dischi può rimanere costante, e la densità di bit decresce dalle tracce interne alle tracce più esterne per mantenere costante la quantità di dati che scorre sotto le testine. Questo metodo si usa nelle unità a disco magnetico ed è noto come **velocità angolare costante** (*constant angular velocity*, CAV).

Il numero di settori per traccia cresce in conseguenza alla tecnologia dei dischi, e l'area più esterna di un disco di solito contiene centinaia di settori per traccia. Anche il numero di cilindri è andato aumentando; le unità a disco contengono decine di migliaia di cilindri.

12.3 Connessione dei dischi

I calcolatori accedono alla memoria secondaria in due modi: nei sistemi di piccole dimensioni il modo più comune è tramite le porte di I/O (**memoria secondaria connessa alla macchina**); oppure ciò avviene in modo remoto per mezzo di un file system distribuito (**memoria secondaria connessa alla rete**).

12.3.1 Memoria secondaria connessa alla macchina

Alla memoria secondaria connessa alla macchina si accede dalle porte locali di I/O. Queste porte sono disponibili in diverse tecnologie; i comuni PC impiegano un'architettura per il bus di I/O chiamata IDE o ATA. Quest'architettura consente di avere non più di due unità per ciascun bus di I/O. Le stazioni di lavoro di fascia alta e i server impiegano architetture più raffinate come SCSI e FC (*fibre channel*).

L'architettura SCSI è un'architettura a bus il cui supporto fisico è di solito un cavo piatto con un gran numero di conduttori (in genere 50 o 68). Consente di avere sul bus fino a 16 dispositivi, comunemente suddivisi in una scheda con il controllore inserita nella macchina (*SCSI initiator*) e in 15 dispositivi di memorizzazione (*SCSI target*), in genere dischi SCSI. Il protocollo permette di accedere fino a 8 **unità logiche** per ciascun dispositivo di memorizzazione. Un uso tipico dell'accesso a unità logiche è l'invio di comandi ai componenti di una batteria RAID, o ai componenti di un archivio di supporti rimovibili (come un *jukebox* di CD che invia comandi al meccanismo per la sostituzione dei CD o a una delle unità).

L'FC è un'architettura seriale ad alta velocità che può funzionare sia su fibra ottica sia su un cavo con 4 conduttori di rame e ha due varianti. La prima è una grande struttura a commutazione con uno spazio d'indirizzi a 24 bit. Per il futuro ci si aspetta che questo metodo prevalga, ed è la base per le **reti di memoria secondaria** (*storage-area networks*, SAN), trattate nel Paragrafo 12.3.3. Grazie al vasto spazio d'indirizzi e alla natura a commutazione della comunicazione, si possono connettere più macchine e dispositivi di memorizzazione alla struttura a commutazione, permettendo una notevole flessibilità nella comunicazione di I/O. La seconda variante si chiama FC-AL (*arbitrated loop*) e può accedere fino a 126 dispositivi (unità e controllori).

C'è un gran numero di dispositivi utilizzabili come memoria secondaria connessa alla macchina, tra questi le unità a disco, le batterie RAID, le unità a CD, DVD e a nastri magnetici. I comandi di I/O che avviano trasferimenti di dati a un dispositivo di memoria connessa alla macchina sono letture e scritture di blocchi logici di dati, dirette a unità di memorizzazione specificamente identificate (ad esempio, tramite bus ID, SCSI ID e unità logica del dispositivo).

12.3.2 Memoria secondaria connessa alla rete

Un dispositivo di memoria secondaria connessa alla rete è un sistema di memoria speciale al quale si accede in modo remoto per mezzo di una rete di trasmissione di dati (Figura 12.2). I client accedono alla memoria connessa alla rete (*network-attached storage*, NAS) tramite un'interfaccia RPC, come l'NFS nel caso dei sistemi UNIX o come CIFS nel caso di sistemi Windows. Le chiamate di procedura remota (RPC) sono realizzate per mezzo dei protocolli TCP o UDP sopra una rete IP (di solito la stessa rete locale che porta tutto il traffico di dati ai client). La memoria secondaria connessa alla rete è normalmente realizzata come una batteria RAID con programmi di controllo che implementano l'interfaccia per le RPC. Conviene

Figura 12.2 Memoria secondaria connessa alla rete.

pensare ai sistemi NAS semplicemente come a un altro protocollo per l'accesso alla memoria secondaria; ad esempio, anziché usare un driver per dispositivi SCSI e i relativi protocolli SCSI per accedere alla memoria secondaria, un sistema che usa sistemi NAS impiega le RPC sopra i protocolli TCP/IP.

La memoria secondaria connessa alla rete fornisce un modo semplice per condividere spazio di memorizzazione a tutti i calcolatori di una LAN, con la stessa facilità di gestione dei nomi e degli accessi caratteristica della memoria secondaria locale. Tuttavia, un sistema di questo genere tende a essere meno efficiente e ad avere prestazioni inferiori rispetto a sistemi che prevedono la connessione diretta alla memoria secondaria.

ISCSI è il più recente protocollo per la memoria connessa alla rete. Essenzialmente sfrutta il protocollo IP della rete per il trasporto del protocollo SCSI. Ne consegue la possibilità di usare cavi di rete invece che cavi SCSI per connettere le diverse macchine alla memoria secondaria. Uno dei vantaggi di questa tecnica è che le macchine sono in grado di trattare la memoria secondaria come se fosse direttamente collegata, sebbene possa essere collocata a distanza.

12.3.3 Reti di memoria secondaria

Uno svantaggio dei sistemi di memoria secondaria connessa alla rete è che le operazioni di I/O sulla memoria secondaria impegnano banda della rete e quindi aumentano la latenza della comunicazione nella rete. Questo problema può essere particolarmente grave per sistemi client-server di grandi dimensioni: l'ordinaria comunicazione tra i server e i client compete per la banda con la comunicazione tra i server e i dispositivi di memorizzazione.

Una rete di memoria secondaria (*storage-area network*, SAN) è una rete privata (che impiega protocolli specifici per la memorizzazione anziché protocolli di rete) tra i server e le unità di memoria secondaria, separata dalla LAN o WAN che collega i server ai client (Figura 12.3). La potenza di una SAN sta nella sua flessibilità: si possono connettere alla stessa SAN molte macchine e molte batterie di memoria; la memoria può essere allocata alle macchine dinamicamente. Uno switch SAN nega o concede alle macchine l'accesso alla memoria secondaria. Per fare un esempio, è possibile configurarlo di modo che allochi memoria secondaria aggiuntiva alle macchine che stanno esaurendo lo spazio sui propri dischi. Questi switch rendono anche possibile la condivisione della memoria di massa da parte di gruppi di server, e il collegamento diretto di più macchine alla memoria di massa. I SAN, infatti, hanno spesso un maggior numero di porte dei dispositivi per la memorizzazione secondaria, a un prezzo più contenuto. Il mezzo più comune per il collegamento tramite SAN è il FC.

Un'alternativa emergente è un'architettura specifica per SAN, chiamata InfiniBand, per reti d'interconnessione ad alta velocità tra server e unità di memoria secondaria.

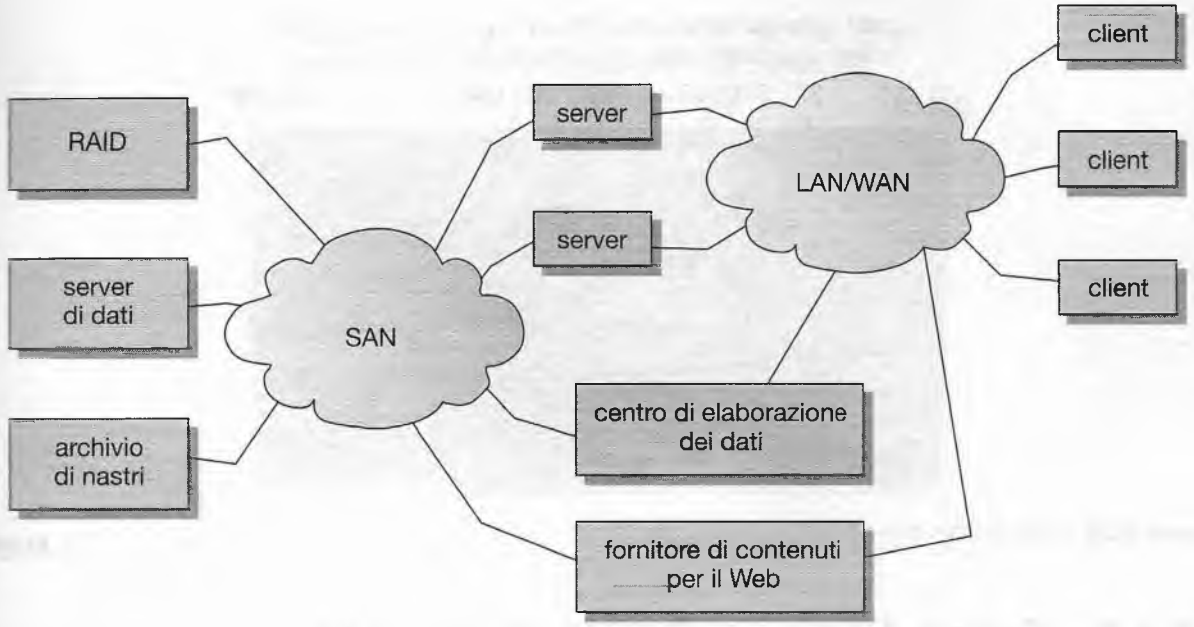


Figura 12.3 Rete di memoria secondaria.

12.4 Scheduling del disco

Una delle responsabilità del sistema operativo è quella di fare un uso efficiente delle risorse fisiche: nel caso delle unità a disco, far fronte a questa responsabilità significa garantire tempi d'accesso contenuti e ampiezze di banda elevate. Il tempo d'accesso si può scindere in due componenti principali (si veda anche il Paragrafo 12.1.1): il **tempo di ricerca** (*seek time*), cioè il tempo necessario affinché il braccio dell'unità a disco sposti le testine fino al cilindro contenente il settore desiderato, e la **latenza di rotazione** (*rotational latency*), e cioè il tempo aggiuntivo necessario perché il disco ruoti finché il settore desiderato si trovi sotto la testina. L'**ampiezza di banda** (*bandwidth*) è il numero totale di byte trasferiti diviso il tempo totale intercorso fra la prima richiesta e il completamento dell'ultimo trasferimento. Per mezzo della gestione dell'ordine delle richieste di I/O relative al disco si possono migliorare sia il tempo d'accesso sia l'ampiezza di banda.

Ogni volta che devono compiere operazioni di I/O con un'unità a disco, un processo impartisce al sistema operativo una chiamata di sistema. La richiesta contiene diverse informazioni:

- se l'operazione sia di immissione o di emissione di dati;
- l'indirizzo nel disco rispetto al quale eseguire il trasferimento;
- l'indirizzo di memoria rispetto al quale eseguire il trasferimento;
- il numero di byte da trasferire.

Se l'unità a disco desiderata e il controllore sono disponibili, la richiesta si può immediatamente soddisfare; altrimenti le nuove richieste si aggiungono alla coda di richieste inevase relativa a quell'unità. La coda relativa a un'unità a disco in un sistema con multiprogrammazione può spesso essere piuttosto lunga, sicché il sistema operativo sceglie quale fra le richieste inevase conviene servire prima. Come il sistema operativo affronta tale scelta? Per mezzo di uno dei tanti algoritmi di scheduling che andiamo a presentare di seguito.

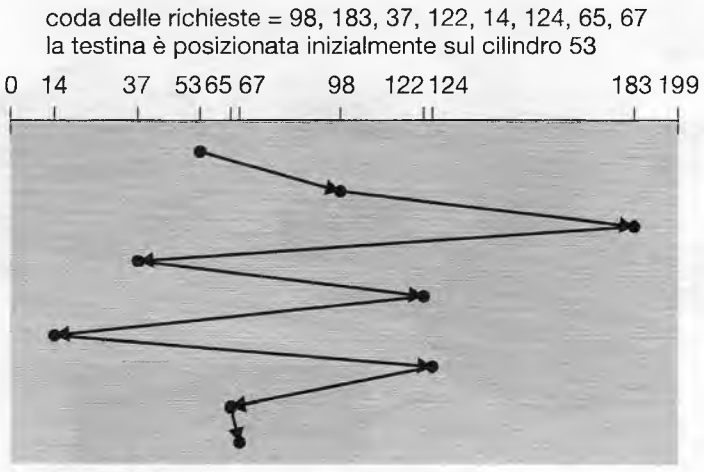


Figura 12.4 Scheduling FCFS.

12.4.1 Scheduling in ordine d'arrivo – FCFS

La forma più semplice di scheduling è, naturalmente, l'algoritmo di servizio secondo l'ordine d'arrivo (*first come, first served*, FCFS). Si tratta di un algoritmo intrinsecamente equo, ma che in generale non garantisce la massima velocità del servizio. Si consideri, ad esempio, una coda di richieste per l'unità a disco che dia una lista di cilindri sui quali individuare i blocchi richiesti, nell'ordine seguente:

98, 183, 37, 122, 14, 124, 65, 67.

Se si trova inizialmente al cilindro 53, la testina dell'unità a disco dovrà prima spostarsi al cilindro 98, poi al 183, 37, 122, 14, 124, 65 e infine al 67, per una distanza totale, misurata in numero di cilindri visitati, di 640 cilindri. La sequenza è rappresentata nella Figura 12.4.

Le deficienze di quest'algoritmo sono palesate dal gigantesco salto da 122 a 14 e poi di nuovo a 124: se le richieste per i cilindri 37 e 14 si potessero soddisfare in sequenza, la distanza totale percorsa diminuirebbe notevolmente e le prestazioni migliorerebbero di conseguenza.

12.4.2 Scheduling per brevità – SSTF

Sembra ragionevole servire tutte le richieste vicine alla posizione corrente della testina prima di spostarla in un'area lontana per soddisfarne altre: questa considerazione è alla base dell'**algoritmo di servizio secondo il più breve tempo di ricerca** (*shortest seek time first*, SSTF), che sceglie la richiesta che dà il minimo tempo di ricerca rispetto alla posizione corrente della testina; poiché questo tempo aumenta al crescere della distanza dei cilindri dalla testina, l'algoritmo sceglie le richieste relative ai cilindri più vicini alla posizione della testina.

Se si considera nuovamente la sequenza di richieste presa ad esempio sopra, il cilindro più vicino alla posizione iniziale della testina (cioè 53) è il 65, la successiva richiesta più vicina è quella relativa al cilindro 67; da questo cilindro, la richiesta relativa al cilindro 37 è più vicina di quella relativa al cilindro 98, ed è quindi servita per terza. Continuando allo stesso modo, sarà soddisfatta la richiesta relativa al cilindro 14, poi 98, 122, 124 e infine 183 (Figura 12.5). Questo metodo di scheduling implica una distanza totale percorsa di soli 236 cilindri, poco più di un terzo della distanza ottenuta con lo scheduling FCFS di questa coda di richieste: esso porta quindi sostanziali miglioramenti d'efficienza.

Lo scheduling SSTF è essenzialmente una forma di scheduling per brevità (*shortest job first*, SJF), e al pari di questo, può condurre a situazioni d'attesa indefinita (*starvation*) di al-

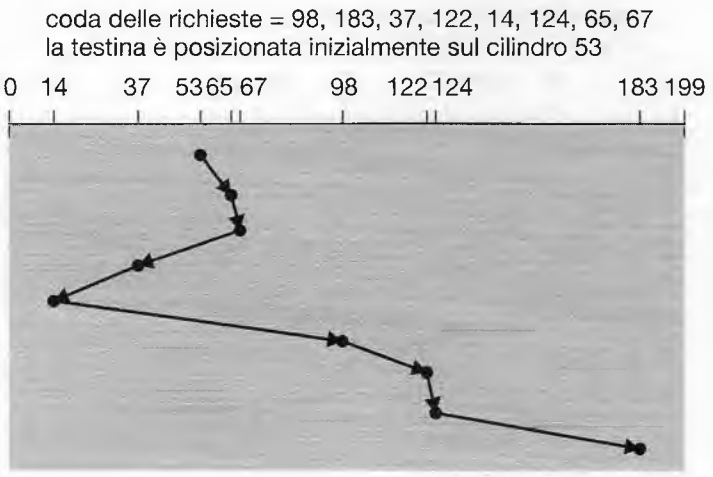


Figura 12.5 Scheduling SSTF.

cune richieste. Si ricordi infatti che nuove richieste possono giungere in qualunque momento: si supponga di avere due richieste in coda, una per il cilindro 14 e l'altra per il 186, e che mentre si sta servendo la richiesta relativa al cilindro 14, arrivi una nuova richiesta per un cilindro vicino al 14. Questa sarà la prossima richiesta soddisfatta, mentre la richiesta per il cilindro 186 dovrà attendere. La stessa situazione potrebbe ripetersi, perché un'altra richiesta relativa a una posizione in prossimità del cilindro 14 potrà giungere mentre si sta servendo la precedente richiesta: in teoria, un flusso continuo di richieste riferite a posizioni le une vicine alle altre potrebbe causare l'attesa indefinita della richiesta relativa al cilindro 186. Quest'ipotesi diviene sempre più probabile man mano che la coda di richieste si allunga.

Sebbene l'algoritmo SSTF costituisca un miglioramento notevole rispetto al FCFS, esso non è ottimale: si può fare di meglio con uno spostamento dal cilindro 53 al 37, anche se questa non è la minima distanza possibile, e poi al 14, prima di invertire la marcia per servire i 65, 67, 98, 122, 124 e 183. L'adozione di questa strategia riduce la distanza totale percorsa a 208 cilindri.

12.4.3 Scheduling per scansione – SCAN

Secondo l'**algoritmo** SCAN il braccio dell'unità a disco parte da un estremo del disco e si sposta nella sola direzione possibile, servendo le richieste mentre attraversa i cilindri, finché non giunga all'altro estremo del disco: a questo punto, il braccio inverte la marcia, e la procedura continua. Le testine attraversano continuamente il disco nelle due direzioni. L'algoritmo SCAN è a volte chiamato **algoritmo dell'ascensore**, perché il braccio dell'unità a disco si comporta proprio come un ascensore che serva prima tutte le richieste in salita e poi tutte quelle in discesa.

Si consideri ancora l'esempio precedente, prima di poter applicare lo scheduling SCAN alle richieste per i cilindri 98, 183, 37, 122, 14, 124, 65, e 67, oltre la posizione corrente (53), occorre conoscere la direzione del movimento delle testine. Se lo spostamento è nella direzione del cilindro 0, l'unità a disco servirà prima la richiesta 37 e poi la 14; una volta giunto al cilindro 0, il braccio invertirà il movimento verso l'altro estremo del disco, servendo le richieste 65, 67, 98, 122, 124 e 183 (Figura 12.6). Se arriva una nuova richiesta riferita a uno dei cilindri posti davanti alla testina (relativamente alla sua direzione di moto), essa sarà quasi immediatamente soddisfatta; ma se la richiesta è riferita a uno dei cilindri appena sorpassati, essa dovrà attendere fino a che la testina non giunga alla fine del disco, inverta la direzione del moto, e torni indietro.

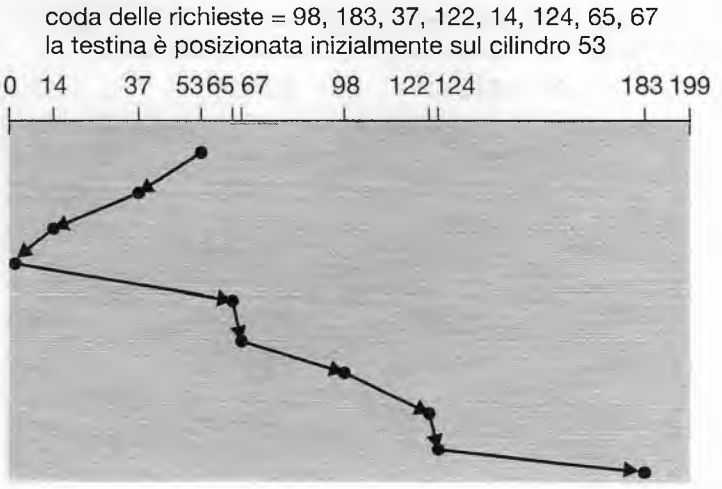


Figura 12.6 Scheduling SCAN.

Assumendo una distribuzione uniforme delle richieste per i cilindri da visitare, si consideri la densità di richieste quando il braccio giunge a un estremo e inverte la direzione del moto: in quel momento, relativamente poche richieste sono riferite a cilindri posti vicino alla testina e davanti a essa, perché i cilindri in questione sono stati recentemente visitati. La massima densità di richieste si riferisce all'altro estremo del disco, e queste richieste sono anche quelle che hanno atteso più a lungo: sembra ragionevole spostarsi lì come prossima mossa. Questa è l'idea dell'algoritmo seguente.

12.4.4 Scheduling per scansione circolare – C-SCAN

L'algoritmo SCAN **circolare** (*circular SCAN*, C-SCAN) è una variante dello scheduling SCAN concepita per garantire un tempo d'attesa meno variabile. Anche l'algoritmo C-SCAN, come lo SCAN, sposta la testina da un estremo all'altro del disco, servendo le richieste lungo il percorso; tuttavia, quando la testina giunge all'altro estremo del disco, ritorna immediatamente all'inizio del disco stesso, senza servire richieste durante il viaggio di ritorno (Figura 12.7). L'algoritmo di scheduling C-SCAN, essenzialmente, tratta il disco come una lista circolare, cioè come se il primo e l'ultimo cilindro fossero adiacenti.

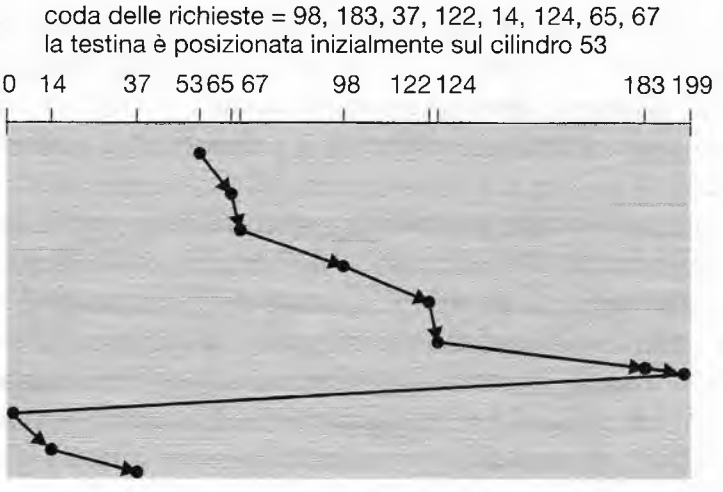


Figura 12.7 Scheduling C-SCAN.

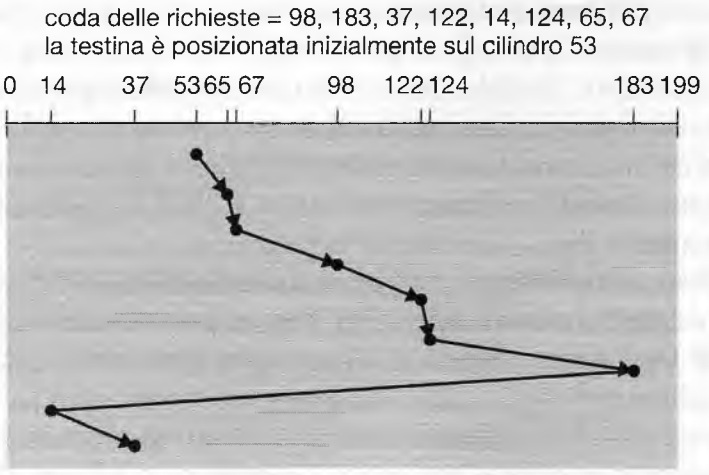


Figura 12.8 Scheduling C-LOOK.

12.4.5 Scheduling LOOK

Secondo la descrizione appena fatta, sia l'algoritmo SCAN sia il C-SCAN spostano il braccio dell'unità attraverso tutta l'ampiezza del disco; all'atto pratico, nessuno dei due algoritmi è codificato in questo modo: più comunemente, il braccio si sposta solo finché ci sono altre richieste da servire in quella direzione, dopo di che cambia immediatamente direzione, senza giungere all'estremo del disco. Queste versioni dello SCAN e del C-SCAN sono dette LOOK e C-LOOK, perché "guardano" (in inglese, *look*) se ci sono altre richieste da soddisfare lungo la direzione attuale prima di continuare a spostare la testina in quella direzione (Figura 12.8).

12.4.6 Scelta di un algoritmo di scheduling

Giacché esistono tanti diversi algoritmi di scheduling, ci si può chiedere come si faccia a scegliere il migliore. Un algoritmo molto comune e naturalmente attraente è l'SSTF poiché aumenta le prestazioni rispetto all'FCFS; lo SCAN e il C-SCAN dànno migliori prestazioni in sistemi che sfruttano molto le unità a disco, perché conducono con minor probabilità a situazioni d'attesa indefinita. Per una data ma arbitraria lista di richieste si può definire un ordine ottimo di servizio, ma la computazione richiesta può non essere giustificata dal miglioramento in prestazioni rispetto agli algoritmi SSTF o SCAN.

Per qualunque algoritmo di scheduling, le prestazioni dipendono comunque in larga misura dal numero e dal tipo di richieste. Ad esempio, si supponga che la coda sia costituita in genere di una sola richiesta inevasa: tutti gli algoritmi dànno allora luogo allo stesso comportamento, perché hanno una sola scelta possibile relativamente al prossimo spostamento della testina. In questo caso, tutti gli algoritmi si comportano come l'FCFS.

Le richieste di I/O per l'unità a disco possono essere notevolmente influenzate dal metodo adottato per l'assegnazione dei file. Un programma che legga un file allocato in modo contiguo genererà molte richieste raggruppate, con un conseguente limitato spostamento della testina. Un file con allocazione concatenata o indicizzata, d'altro canto, potrebbe includere blocchi sparsi per tutto il disco, e richiedere quindi un maggiore movimento della testina.

Anche la posizione delle directory e dei blocchi indice è importante: poiché ogni file deve essere aperto per essere usato, e visto che l'apertura di un file richiede una ricerca attraverso la struttura delle directory, vi saranno frequenti accessi alle directory. Si supponga che

un elemento di directory risieda nel primo cilindro e che i dati del file relativo si trovino nell'ultimo cilindro; la testina dovrà allora percorrere l'intera ampiezza del disco nel caso di apertura del file in questione. Se l'elemento della directory che rappresenta logicamente il file fosse nel cilindro di mezzo, la testina dovrebbe spostarsi al più della metà dell'ampiezza. Anche l'uso della memoria centrale come cache delle directory e dei blocchi indice può contribuire a ridurre i movimenti del braccio dell'unità a disco, in particolare quando si tratta di operazioni di lettura.

A causa di queste complicazioni, l'algoritmo di scheduling del disco dovrebbe costituire un modulo a sé stante del sistema operativo, così da poter essere sostituito da un altro algoritmo qualora ciò fosse necessario; come algoritmo di partenza è ragionevole la scelta dell'SSTF o del LOOK.

Gli algoritmi di scheduling descritti tengono conto solamente del tempo di ricerca, mentre nelle moderne unità a disco la latenza di rotazione può essere lunga quasi quanto il tempo medio di ricerca. Tuttavia è difficile per il sistema operativo adottare una strategia di scheduling che porti a miglioramenti dei tempi di latenza di rotazione, perché le moderne unità a disco non rivelano la posizione fisica dei blocchi logici. I produttori di unità a disco hanno collaborato alla limitazione di questo problema incorporando algoritmi di scheduling all'interno dei controllori contenuti nelle unità a disco: se il sistema operativo invia un gruppo di richieste al controllore, esso può organizzarle in una coda e poi applicare algoritmi di scheduling che riducano sia i tempi di ricerca sia la latenza di rotazione.

Se l'unico elemento di cui tener conto fossero le prestazioni dell'I/O, il sistema operativo scaricherebbe volentieri la responsabilità dello scheduling per il disco sull'apparato elettronico del dispositivo. In pratica, però, il sistema operativo può dover considerare altri vincoli relativi all'ordine in cui si devono servire le richieste: ad esempio, la richiesta di una pagina di memoria virtuale potrebbe avere maggiore priorità rispetto all'I/O delle applicazioni, e le scritture divengono più urgenti delle letture quando la cache sta per esaurire le pagine disponibili. Inoltre, può essere auspicabile mantenere l'ordine naturale delle richieste di scrittura al fine di rendere il file system robusto rispetto ai crolli del sistema: si consideri cosa accadrebbe se il sistema operativo assegnasse un blocco di disco a un file, e un'applicazione scrivesse dati in quel blocco; se il sistema crollasse a questo punto, il sistema operativo potrebbe non essere riuscito a copiare l'*inode* modificato e la nuova lista dello spazio libero sul disco. Per conciliare queste esigenze, il sistema operativo può scegliere di accollarsi la responsabilità dello scheduling del disco e, per alcuni tipi di I/O, fornire le richieste al controllore una alla volta.

12.5 Gestione dell'unità a disco

Il sistema operativo è anche responsabile di molti altri aspetti della gestione delle unità a disco. In questo paragrafo si discutono l'inizializzazione del disco, l'avviamento del sistema basato sull'unità a disco, e la gestione dei blocchi difettosi.

12.5.1 Formattazione del disco

Un disco magnetico nuovo è *tabula rasa*: un insieme di uno o più piatti sovrapposti ricoperti di materiale magnetico; prima che possa memorizzare dati, deve essere diviso in settori che possano essere letti o scritti dal controllore. Questo processo si chiama formattazione di basso livello, o **formattazione fisica**. La **formattazione di basso livello** riempie il disco con una speciale struttura dati per ogni settore, tipicamente consistente di un'intestazione, un'area per i dati (di solito di 512 byte), e una coda. L'intestazione e la coda contengono in-

formazioni usate dal controllore del disco, ad esempio il numero del settore e un **codice per la correzione degli errori** (*error-correcting code*, ECC). Quando il controllore scrive dati in un settore nel corso di un'ordinaria operazione di I/O, aggiorna il valore dell'ECC secondo il contenuto dell'area di dati del settore. Quando il controllore legge dati da quel settore, calcola anche l'ECC e lo confronta con il suo valore memorizzato: se risulta una discrepanza, l'area dei dati del settore non è integra, e il settore del disco potrebbe essere difettoso (si veda il Paragrafo 12.5.3). L'ECC è un codice per la *correzione* degli errori: se solo alcuni bit di dati sono stati alterati, esso contiene sufficienti informazioni affinché il controllore possa identificare i bit in questione e ricalcolare il loro corretto valore. Il controllore esegue automaticamente l'elaborazione descritta ogni volta che accede a un settore del disco.

La formattazione fisica dei dischi è eseguita nella maggior parte dei casi dal costruttore come parte del processo produttivo; ciò permette al costruttore di provare il disco, e di instaurare la corrispondenza fra blocchi logici e settori correttamente funzionanti del disco. In molte unità a disco, quando si richiede al controllore di formattare fisicamente il disco, si può anche specificare il numero di byte delle aree di dati comprese fra l'intestazione e la coda di un settore. La scelta è di solito ristretta a poche opzioni, come 256, 512 o 1024 byte. La formattazione in settori più grandi implica la presenza di meno settori su ogni traccia, ma anche meno intestazioni e code, e quindi maggior spazio per i dati veri e propri. Alcuni sistemi operativi gestiscono solo settori di 512 byte.

Per usare un disco come contenitore d'informazioni, il sistema operativo deve ancora registrare le proprie strutture dati nel disco, cosa che fa in due passi. Il primo consiste nel suddividere il disco in uno o più gruppi di cilindri, detti **partizioni**. Il sistema operativo può trattare ogni gruppo come se fosse un'unità a disco a sé stante: ad esempio, una partizione può contenere una copia del codice eseguibile del sistema operativo, mentre un'altra contiene i file degli utenti. Il passo successivo alla suddivisione in partizioni è la **formattazione logica**, cioè la creazione di un file system: il sistema operativo registra nel disco le strutture dati iniziali relative al file system. Le strutture dati in questione possono includere descrizioni dello spazio libero e dello spazio allocato (FAT o *inode*) e una directory iniziale vuota.

Per una maggior efficienza, molti file system accorpano i blocchi in gruppi, detti cluster. L'I/O del disco procede per blocchi, ma l'I/O del file system procede invece per cluster, di modo che l'I/O mutui sempre più caratteristiche dall'accesso sequenziale che non dall'accesso casuale.

Alcuni sistemi operativi dànno l'opportunità a certi programmi speciali di impiegare una partizione del disco come un grande array sequenziale di blocchi logici, non contenente alcuna struttura dati relativa al file system, detto disco di basso livello (*raw disk*); l'I/O relativo si chiama I/O di basso livello (*raw I/O*). Alcuni sistemi per la gestione di basi di dati, ad esempio, preferiscono questo tipo di I/O perché permette di controllare l'esatta posizione nel disco d'ogni informazione trattata. Esso scavalca tutti i servizi del file system: gestione delle *buffer cache*, prelievo anticipato, allocazione dello spazio, nomi dei file, directory, e così via. È possibile rendere certe applicazioni più efficienti includendovi servizi di memorizzazione secondaria specializzati che usino una partizione di basso livello, ma la maggior parte delle applicazioni è in realtà migliore quando usufruisce degli ordinari servizi del file system.

12.5.2 Blocco d'avviamento

Affinché un calcolatore possa entrare in funzione, ad esempio quando viene acceso o riavviato, è necessario che esegua un programma iniziale; di solito, questo programma d'avviamento iniziale è piuttosto semplice. Esso inizializza il sistema in tutti i suoi aspetti, dai registri della CPU ai controllori dei dispositivi e al contenuto della memoria centrale, quindi avvia il sistema operativo. Per far ciò, il programma d'avviamento trova il kernel del sistema

operativo nei dischi, lo carica nella memoria, e salta a un indirizzo iniziale per avviare l'esecuzione del sistema operativo.

Per la maggior parte dei calcolatori, il programma d'avviamento è memorizzato in una **memoria a sola lettura** (*read-only memory*, ROM), il che è conveniente, perché la ROM non richiede inizializzazione, e ha un indirizzo iniziale fisso dal quale la CPU può cominciare l'esecuzione ogniqualvolta si accende o si riavvia la macchina. Inoltre, visto che la ROM è a sola lettura, non può essere contaminata da un virus informatico. Il problema, però, è che cambiare il programma d'avviamento richiede in questo caso la sostituzione dei circuiti integrati ROM. A causa di questo inconveniente, molti sistemi memorizzano nella ROM un piccolo caricatore d'avviamento (*bootstrap loader*) il cui solo compito è quello di caricare da un disco il programma d'avviamento completo. Quest'ultimo si può facilmente modificare: se ne scrive semplicemente una nuova versione nel disco. Il programma d'avviamento completo è registrato in una partizione del disco denominata blocchi d'avviamento, posta in una locazione fissata del disco; un disco contenente una tale partizione si chiama **disco d'avviamento** o **disco di sistema**.

Il codice contenuto nella ROM d'avviamento istruisce innanzi tutto il controllore dell'unità a disco affinché trasferisca il contenuto dei blocchi d'avviamento nella memoria (si noti che a questo fine non si carica alcun driver di dispositivo), quindi comincia a eseguire il codice. Il programma d'avviamento completo è più complesso del suo caricatore, ed è capace di trasferire nella memoria l'intero sistema operativo inizialmente residente in un disco in una locazione non definitivamente fissata, e di avviare il sistema operativo stesso. Il programma d'avviamento potrà comunque essere breve.

Consideriamo come esempio il processo d'avviamento in Windows 2000. Questo sistema colloca il proprio codice d'avviamento nel primo settore del disco rigido, denominato MBR (*master boot record*). Esso, inoltre, consente di suddividere il disco rigido in una o più partizioni; in una di loro, detta **partizione d'avviamento**, sono contenuti il sistema operativo e i driver dei dispositivi. La procedura d'avviamento di Windows 2000 inizia con l'esecuzione del codice residente nella memoria ROM del sistema. Questa parte del codice guida il sistema a leggere il codice d'avviamento dall'MBR. Oltre al codice d'avviamento, l'MBR contiene una tabella che elenca le partizioni del disco rigido e un flag che indica da quale partizione si debba avviare il sistema; ciò è illustrato nella Figura 12.9. Dopo aver identificato la partizione d'avviamento, il sistema legge da tale partizione il primo settore (chiamato **settore d'avviamento**) e svolge le restanti procedure d'avviamento, tra cui il caricamento dei vari sottosistemi e dei servizi del sistema.

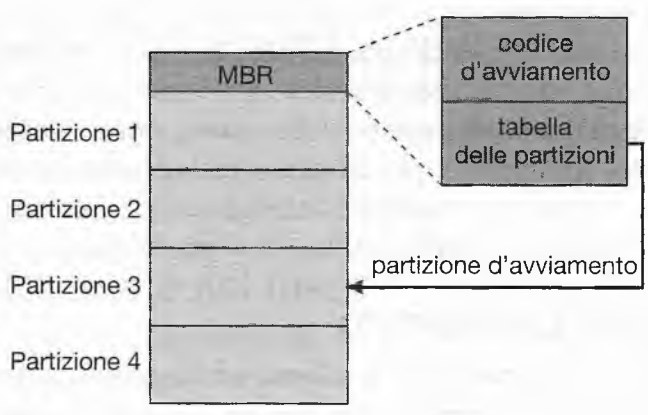


Figura 12.9 Avviamento dal disco di Windows 2000.

12.5.3 Blocchi difettosi

Le unità a disco sono strutturalmente portate ai malfunzionamenti perché sono costituite da parti mobili a bassa tolleranza (si ricordi che una testina è sospesa appena sopra la superficie del disco). A volte si può verificare un guasto irreparabile, e l'unità a disco deve essere sostituita: le informazioni contenute nel disco dovranno essere recuperate da una copia di riserva mantenuta separatamente e trasferite nella nuova unità a disco. Più di frequente, uno o più settori divengono malfunzionanti; in effetti, la maggior parte dei dischi messi in commercio contiene già **blocchi difettosi**. Essi sono trattati in diversi modi secondo il controllore e l'unità a disco presenti nel sistema.

Nel caso di dischi semplici come quelli gestiti da un controllore IDE, i blocchi difettosi sono gestiti "manualmente". Ad esempio, il comando `format` dell'MS-DOS esegue una formattazione logica, e come parte del processo esamina il disco per rilevare la presenza di blocchi difettosi: se ne trova qualcuno, scrive un valore speciale nel corrispondente elemento della FAT al fine di segnalare alle procedure di allocazione di non usare il blocco in questione. Se qualche blocco diviene malfunzionante nel corso dell'ordinario uso del sistema, un programma speciale (ad esempio `chkdsk`) deve essere eseguito a cura dell'utente per individuare i blocchi difettosi e isolarli come appena descritto. Di solito, i dati residenti nei blocchi difettosi vanno perduti.

Unità a disco più complesse come i dischi SCSI in uso nei PC di alto livello e nella maggior parte delle stazioni di lavoro e server, hanno strategie di recupero dei blocchi difettosi più raffinate. Il controllore mantiene una lista dei blocchi malfunzionanti dell'unità a disco che è inizializzata durante la formattazione fisica eseguita dal produttore, ed è aggiornata per tutto il periodo in cui l'unità a disco è operativa. La formattazione fisica mette anche da parte dei settori di riserva non visibili al sistema operativo: si può istruire il controllore affinché sostituisca da un punto di vista logico un settore difettoso con uno dei settori di riserva inutilizzati. Questa strategia è nota come **accantonamento di settori** (*sector sparing* o *sector forwarding*).

Un tipico esempio di attuazione di questa strategia è il seguente:

- il sistema operativo tenta di leggere il blocco logico 87;
- il controllore calcola l'ECC e scopre che il settore è difettoso; quindi segnala questo malfunzionamento al sistema operativo;
- la volta successiva che il sistema viene riavviato, si esegue un comando speciale al fine di comunicare al controllore SCSI la necessità di sostituire il settore difettoso con uno di riserva;
- dopo di ciò, ogni volta che il sistema tenta di leggere il contenuto del blocco 87, il controllore traduce la richiesta nell'indirizzo del settore di rimpiazzo.

Un tale reindirizzamento da parte del controllore potrebbe inficiare ogni ottimizzazione fornita dall'algoritmo di scheduling del disco del sistema operativo. Per questa ragione la maggior parte dei dischi si formatta in modo tale da mantenere qualche settore di riserva in ogni cilindro, e anche un intero cilindro di riserva. Quando un numero di blocco logico è allocato a dei settori di riserva, il controllore usa settori di riserva presenti nello stesso cilindro ogniqualvolta ciò sia possibile.

Un'alternativa all'accantonamento dei settori è data da quei controllori capaci di sostituire i settori difettosi tramite la tecnica della **traslazione dei settori** (*sector slipping*). Si supponga ad esempio che il blocco logico 17 divenga malfunzionante, e che il primo settore di riserva disponibile sia quello successivo al settore 202. La traslazione dei settori sposterebbe in avanti di un posto tutti i settori dal 17 al 202: quindi, il settore 202 sarebbe copiato sul

settore di riserva, il settore 201 sul 202, il 200 sul 201, e così via, fino a che il settore 18 non sia stato copiato sul 19. Questa traslazione dei settori libera lo spazio del settore 18, e il settore 17 può essere fatto corrispondere a quest'ultimo.

La sostituzione di un blocco difettoso non è in genere un processo totalmente automatico, perché i dati contenuti nel blocco in questione di solito vanno perduti. Un file che usava quel blocco deve quindi essere riparato (ad esempio, ricopiandolo da un nastro contenente le copie di riserva), e ciò richiede un intervento manuale.

12.6 Gestione dell'area d'avvicendamento

L'avvicendamento (*swapping*) è stato introdotto, inizialmente, nel Paragrafo 8.2, dove abbiamo trattato lo spostamento di interi processi tra disco e memoria centrale. In quel contesto, l'avvicendamento interviene quando l'ammontare della memoria fisica si abbassa fino al punto di raggiungere la soglia critica e i processi (solitamente scelti in quanto meno attivi) passano dalla memoria all'area d'avvicendamento, per liberare memoria. Nella pratica, pochissimi sistemi operativi moderni realizzano l'avvicendamento nel modo descritto: essi, infatti, combinano l'avvicendamento con tecniche di memoria virtuale (Capitolo 9) per coinvolgere nell'operazione solo alcune pagine, e non necessariamente interi processi. Tant'è che alcuni sistemi considerano *avvicendamento* e *paginazione* termini intercambiabili, a riprova della moderna tendenza a convergere di questi due concetti.

La **gestione dell'area d'avvicendamento** è un altro compito di basso livello del sistema operativo. La memoria virtuale usa lo spazio dei dischi come estensione della memoria centrale: poiché l'accesso alle unità a disco è molto più lento dell'accesso alla memoria centrale, l'uso di un'area d'avvicendamento riduce notevolmente le prestazioni del sistema. L'obiettivo principale nella progettazione e realizzazione di un'area d'avvicendamento è di fornire la migliore produttività per il sistema della memoria virtuale. In questo paragrafo sono trattati l'uso, la collocazione nei dischi e la gestione dell'area d'avvicendamento.

12.6.1 Uso dell'area d'avvicendamento

L'area d'avvicendamento è usata in modi diversi da sistemi operativi diversi, in funzione degli algoritmi di gestione della memoria applicati. I sistemi che adottano l'avvicendamento dei processi nella memoria, ad esempio, possono usare l'area d'avvicendamento per mantenere l'intera immagine del processo, inclusi i segmenti dei dati e del codice; i sistemi a paginazione, invece, possono semplicemente memorizzarvi pagine non contenute nella memoria centrale. Lo spazio richiesto dall'area d'avvicendamento per un sistema può quindi variare secondo la quantità di memoria fisica, la quantità di memoria virtuale che esso deve sostenere, e il modo in cui quest'ultima è usata.

Si noti che una stima per eccesso delle dimensioni dell'area d'avvicendamento è più prudente di una per difetto, perché un sistema che esaurisca l'area d'avvicendamento potrebbe essere costretto a terminare forzatamente i processi o ad arrestarsi completamente: una stima per eccesso spreca spazio dei dischi che si potrebbe usare per i file, ma non provoca altri danni. Alcuni sistemi consigliano la quantità da stanziare per l'area d'avvicendamento. Solaris, ad esempio, raccomanda di riservare a tal fine uno spazio uguale alla quantità di memoria virtuale che eccede la memoria fisica paginabile. Linux ha sempre suggerito di raddoppiare l'area d'avvicendamento rispetto alla memoria fisica, quantunque molti sistemi Linux usino attualmente un'area d'avvicendamentonotevolmente minore. Addirittura, al-

l'interno della comunità Linux, è vivacemente discussa la mera opportunità di destinare o meno un'area all'avvicendamento!

Alcuni sistemi operativi, fra i quali Linux, permettono l'uso di aree d'avvicendamento multiple, poste di solito in unità a disco distinte per distribuire su più dispositivi il carico della paginazione e dell'avvicendamento dei processi gravante sul sistema per l'I/O.

12.6.2 Collocazione dell'area d'avvicendamento

Le possibili collocazioni per un'area d'avvicendamento sono due: all'interno del normale file system, o in una partizione del disco a sé stante. Se l'area d'avvicendamento è semplicemente un grande file all'interno del file system, si possono usare le ordinarie funzioni del file system per crearla, assegnargli un nome, e allocare spazio per essa. Questo criterio sebbene sia semplice da realizzare è inefficiente: l'attraversamento della struttura delle directory e l'uso delle strutture dati per l'allocazione dello spazio nei dischi richiede tempo, oltre che, almeno potenzialmente, accessi ai dischi aggiuntivi. La frammentazione esterna può aumentare molto i tempi d'avvicendamento causando ricerche multiple durante la scrittura o la lettura dell'immagine di un processo. Le prestazioni si possono migliorare impiegando la memoria fisica come cache per le informazioni relative alla posizione dei blocchi, e anche usando strumenti speciali per l'allocazione in blocchi fisicamente contigui del file d'avvicendamento, ma il costo dovuto all'attraversamento del file system e delle sue strutture dati permane.

In alternativa, l'area d'avvicendamento si può creare in un'apposita partizione del disco non formattata: in essa non è presente alcuna struttura relativa al file system e alle directory, ma si usa uno speciale gestore dell'area d'avvicendamento per allocare e rimuovere i blocchi. Esso adotta algoritmi ottimizzati rispetto alla velocità, e non rispetto allo spazio impiegato, dato che all'area d'avvicendamento (quando viene utilizzata) si accede molto più frequentemente che ai file system. La frammentazione interna può aumentare, ma questo prezzo da pagare è ragionevole perché i dati nell'area d'avvicendamento hanno una vita media molto più breve dei file ordinari, e gli accessi all'area d'avvicendamento sono in genere molto più frequenti. Poiché l'area d'avvicendamento è inizializzata all'avvio, la frammentazione ha vita breve. Questo metodo assegna una dimensione fissa all'area d'avvicendamento al momento della creazione delle partizioni del disco, e l'aumento delle dimensioni dell'area d'avvicendamento deve quindi passare attraverso il ripartizionamento del disco (che implica lo spostamento o l'eliminazione e la sostituzione con copie di riserva delle altre partizioni del disco), o attraverso la creazione di un'altra area d'avvicendamento in qualche altra unità a disco del sistema.

Alcuni sistemi operativi non adottano una strategia rigida e possono costruire aree d'avvicendamento sia su partizioni specifiche sia all'interno del file system: Linux ne è un esempio. I metodi di gestione e la loro concreta realizzazione sono diversi nei due casi, e la scelta fra le due soluzioni è lasciata all'amministratore del sistema: sul piatto della bilancia pesano da un lato la convenienza dell'allocazione e della gestione dell'area d'avvicendamento all'interno del file system, e dall'altro le migliori prestazioni ottenibili grazie all'uso di una partizione specifica.

12.6.3 Gestione dell'area d'avvicendamento: un esempio

Si può comprendere come l'area d'avvicendamento venga applicata ripercorrendo la sua evoluzione e quella della paginazione nei vari sistemi UNIX. Il kernel tradizionale di UNIX implementava inizialmente una tecnica d'avvicendamento che spostava interi processi tra le regioni contigue del disco e la memoria. Più tardi, con la disponibilità dei dispositivi per la paginazione, UNIX progredì verso una combinazione d'avvicendamento e paginazione.

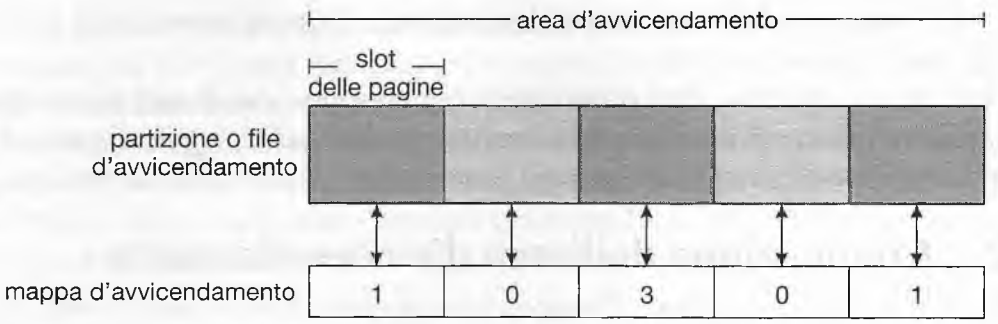


Figura 12.10 Strutture dati per l'avvicendamento nei sistemi Linux.

Per migliorare l'efficienza e assimilare le innovazioni tecnologiche, in Solaris 1 (SunOS) i progettisti cambiarono le tecniche tradizionali di UNIX. Quando un processo va in esecuzione, le pagine contenenti il codice eseguibile sono prelevate dal file system, poste in memoria centrale e, qualora fossero state selezionate per l'espulsione, eliminate. Rileggere una pagina dal file system è più efficiente che scriverla nell'area d'avvicendamento e rileggerla da lì: l'area d'avvicendamento è adoperata esclusivamente come area di stoccaggio per le pagine della **memoria anonima**, di cui fanno parte la memoria allocata per la pila, quella per lo heap e i dati non inizializzati dei processi.

Ulteriori cambiamenti sono stati introdotti nelle versioni più recenti di Solaris. Il più importante prevede che l'area d'avvicendamento sia allocata solo quando una pagina è espulsa dalla memoria fisica, anziché quando la pagina è creata per la prima volta in memoria virtuale. Questa regola produce un miglioramento delle prestazioni nei calcolatori moderni, i quali, rispetto ai sistemi più vecchi, possono contare su una maggiore quantità di memoria fisica e limitare il ricorso alla paginazione.

Così come per Solaris, l'area d'avvicendamento di Linux è utilizzata soltanto per la memoria anonima o per regioni di memoria condivise tra numerosi processi. Linux permette di istituire una o più aree d'avvicendamento, sia in file di avvicendamento (*swap file*) del file system regolare, sia in una partizione a basso livello dedicata all'area di avvicendamento (*swap area*). Un'area d'avvicendamento è formata da una serie di moduli di 4 KB detti **slot delle pagine**, la cui funzione è di conservare le pagine scambiate. Ogni area d'avvicendamento ha una **mappa d'avvicendamento** (*swap map*) associata, ovvero un array di contatori interi, ciascuno dei quali corrisponde a uno slot nell'area d'avvicendamento. Se un contatore segna il valore 0, la pagina che gli corrisponde è disponibile. Valori superiori a 0 indicano che lo slot è occupato da una delle pagine scambiate. Il valore del contatore indica il numero di collegamenti alla pagina scambiata; se esso è 3, per esempio, allora la pagina scambiata è associata a tre processi differenti, eventualità possibile nel caso che la pagina rappresenti una regione di memoria condivisa da tre processi. Le strutture dati relative all'avvicendamento nei sistemi Linux sono rappresentate dalla Figura 12.10.

12.7 Strutture RAID

L'evoluzione tecnologica ha reso le unità a disco progressivamente più piccole e meno costose, tanto che oggi è possibile, senza eccessivi sforzi economici, equipaggiare un sistema di calcolo con molti dischi. La presenza di più dischi, qualora si possano usare in parallelo, rende possibile l'aumento della frequenza a cui i dati si possono leggere o scrivere. Inoltre, una

STRUTTURA DEI DISPOSITIVI RAID

Le memorie di massa RAID si prestano a essere strutturate con modalità diverse. Un sistema, ad esempio, può collegare direttamente i dischi ai propri bus, nel qual caso la funzionalità RAID può essere realizzata dal sistema operativo o dai programmi di sistema. In alternativa, un controllore intelligente può gestire diversi dischi collegati alla stessa macchina e implementare una struttura RAID per quei dischi a livello hardware. Infine, si può ricorrere a una **batteria** RAID (*RAID array*), un'unità a sé stante, dotata di un controllore, di una cache (nella maggioranza dei casi) e di dischi autonomi. La batteria è collegata alla macchina attraverso uno o più controllori standard ATA SCSI o FC. Questa diffusa organizzazione consente a programmi e sistemi operativi di per sé privi della funzionalità RAID di usufruirne comunque. Persino sistemi che, in realtà, implementano le funzionalità RAID a livello software adottano a volte la tecnica descritta, per via della sua semplicità e flessibilità.

configurazione di questo tipo permette di migliorare l'affidabilità della memoria secondaria, poiché diventa possibile memorizzare le informazioni in più dischi in modo ridondante. In questo caso, un guasto a uno dei dischi non comporta la perdita di dati. Ci sono varie tecniche per l'organizzazione dei dischi, note col nome comune di **batterie ridondanti di dischi** (*redundant array of independent* [*inexpensive*] *disks*, RAID), che hanno lo scopo di affrontare i problemi di prestazioni e affidabilità.

Nel passato, strutture RAID composte da piccoli dischi economici erano viste come un'alternativa economicamente vantaggiosa rispetto a costosi dischi di grande capacità; oggi, le strutture RAID s'impiegano per la loro maggiore affidabilità e velocità di trasferimento dei dati, piuttosto che per ragioni economiche. Quindi, la *I* in RAID attualmente andrebbe letta *independent* anziché *inexpensive* com'era interpretata originariamente.

12.7.1 Miglioramento dell'affidabilità tramite la ridondanza

Consideriamo in primo luogo l'affidabilità. La possibilità che uno dei dischi in un insieme di n dischi si guasti è molto più alta della possibilità che uno specifico disco isolato presenti un guasto. Si supponga che il **tempo medio di guasto** di un singolo disco sia 100.000 ore. In questo caso, il tempo medio di guasto per un qualsiasi disco in una batteria di 100 dischi sarebbe 100.000/100 = 1000 ore, o 41,66 giorni; cioè non molto tempo. Se si memorizzasse una sola copia dei dati, allora ogni guasto di un disco comporterebbe la perdita di una notevole quantità di dati; una frequenza di perdita di dati così alta sarebbe inaccettabile.

La soluzione al problema dell'affidabilità sta nell'introdurre una certa **ridondanza**, cioè nel memorizzare informazioni che non sono normalmente necessarie, ma che si possono usare nel caso di un guasto a un disco per ricostruire le informazioni perse.

Il metodo più semplice (ma anche il più costoso) di introduzione di ridondanza è quello della **copiatura speculare** (*mirroring* o *shadowing*); ogni disco logico consiste di due dischi fisici e ogni scrittura si effettua in entrambi i dischi. Se uno dei dischi si guasta, i dati si possono leggere dall'altro. I dati si perdono solo se il secondo disco si guasta prima della sostituzione del disco già guasto.

Il tempo medio di guasto di un disco con copiatura speculare, dove per *guasto* s'intende ora la perdita di dati, dipende da due fattori: il tempo medio di guasto di un singolo disco e il **tempo medio di riparazione**, cioè il tempo richiesto (in media) per sostituire un disco guasto e ripristinarvi i dati. Supponendo che i possibili guasti dei due dischi siano **indi-**

pendenti, vale a dire che il guasto di un disco non sia mai legato a quello dell'altro, se il tempo medio di guasto di un singolo disco è 100.000 ore e il tempo medio di riparazione è di 10 ore, allora il tempo medio di perdita di dati di un sistema con copiatura speculare dei dischi è $100.000^2/(2 \times 10) = 500 \times 10^6$ ore, che corrispondono a 57.000 anni!

Occorre però notare che l'ipotesi di indipendenza tra i guasti dei dischi non è in realtà valida, poiché improvvisi cali di tensione e disastri naturali, quali terremoti, incendi e alluvioni, danneggerebbero con tutta probabilità entrambi i dischi. Inoltre, difetti di fabbricazione in una partita di dischi possono causare guasti simili e correlati. Con l'invecchiamento del disco, la probabilità di un guasto aumenta, accrescendo la probabilità che un secondo disco si guasti mentre il primo è in riparazione. Tuttavia, nonostante tutte queste considerazioni, i sistemi con copiatura speculare dei dischi offrono un'affidabilità assai più alta dei sistemi a disco singolo.

I casi di improvvisa mancanza di tensione elettrica costituiscono un problema particolarmente sentito, poiché avvengono con una frequenza molto più alta dei disastri naturali. Tuttavia, anche impiegando la copiatura speculare dei dischi, se si sta svolgendo un'operazione di scrittura nello stesso blocco in entrambi i dischi e si verifica una mancanza di tensione prima che sia completata la scrittura dell'intero blocco, i due blocchi possono ritrovarsi in uno stato incoerente. Una soluzione prevede la scrittura di una delle due copie e solo successivamente la scrittura della seconda, così che una delle due copie sia sempre coerente. Un'altra soluzione è aggiungere una memoria cache non volatile (NVRAM, *non-volatile RAM*) alla batteria RAID, protetta dalla perdita di dati causata dalle cadute di tensione; se è dotata di forme di correzione d'errore come ECC o copiatura speculare i dati nella cache possono essere considerati completi.

12.7.2 Miglioramento delle prestazioni tramite il parallelismo

L'accesso in parallelo a più dischi può portare vari vantaggi. Con la copiatura speculare dei dischi, la frequenza con la quale si possono gestire le richieste di lettura raddoppia, poiché ciascuna richiesta si può inviare indifferentemente a uno dei due dischi (sempre che entrambi i dischi siano funzionanti, condizione che è quasi sempre soddisfatta). La capacità di trasferimento di ciascuna lettura è la stessa di quella di un sistema a singolo disco, ma il numero di letture per unità di tempo raddoppia.

Attraverso l'uso di più dischi è possibile anche (o alternativamente) migliorare la capacità di trasferimento distribuendo i dati in sezioni su più dischi. Nella sua forma più semplice questa distribuzione, chiamata **sezionamento dei dati** (*data striping*), consiste nel distribuire i bit di ciascun byte su più dischi; in questo caso si parla di **sezionamento a livello dei bit**. Ad esempio, se il sistema impiega una batteria di otto dischi, si scriverà il bit i di ciascun byte nel disco i. La batteria di otto dischi si può trattare come un unico disco avente settori che hanno una dimensione otto volte superiore a quella normale e, soprattutto, che hanno una capacità di trasferimento otto volte superiore. In un'organizzazione di questo tipo, ogni disco è coinvolto in ogni accesso (lettura o scrittura che sia), così che il numero di accessi che si possono gestire nell'unità di tempo è circa lo stesso di quello per un sistema a disco singolo, ma ogni accesso permette di leggere una quantità di dati pari a otto volte quella che si può leggere con un singolo disco.

Il sezionamento a livello dei bit si può generalizzare a un numero di dischi multiplo di 8 o che divide 8. Ad esempio, se un sistema adopera una batteria di quattro dischi, i bit i e $i + 4$ di ciascun byte si memorizzano nel disco i. Inoltre, il sezionamento non si deve realizzare necessariamente a livello dei bit di un byte: nel **sezionamento a livello dei blocchi**, ad

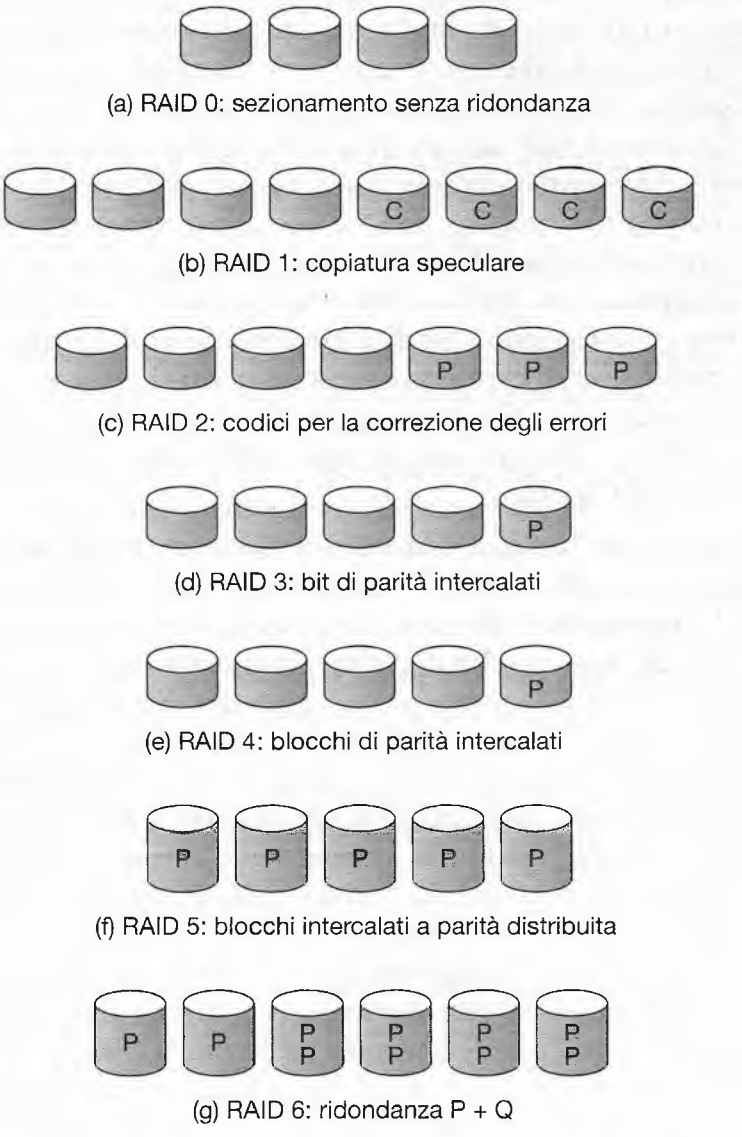
esempio, i blocchi di un file si distribuiscono su più dischi; con n dischi, il blocco i di un file si memorizza nel disco (i mod n) + 1. Sono possibili anche altri livelli di sezionamento, come quelli basati sui byte di un settore o sui settori di un blocco.

Riassumendo, gli obiettivi principali riguardo al parallelismo in un sistema di dischi sono due:

1. l'aumento, tramite il bilanciamento del carico, della produttività per accessi multipli a piccole porzioni di dati (cioè accessi a pagine);
2. la riduzione del tempo di risposta relativo ad accessi a grandi quantità di dati.

12.7.3 Livelli RAID

La tecnica di copiatura speculare offre un'alta affidabilità ma è costosa; la tecnica del sezionamento offre un'alta capacità di trasferimento dei dati, ma non migliora l'affidabilità. Sono stati proposti numerosi schemi per fornire ridondanza usando l'idea del sezionamento combinata con i bit di parità. Questi schemi realizzano diversi compromessi tra costi e prestazioni e sono stati classificati in livelli chiamati **livelli** RAID, che la Figura 12.11 mostra



(a) RAID 0: sezionamento senza ridondanza

(b) RAID 1: copiatura speculare

(c) RAID 2: codici per la correzione degli errori

(d) RAID 3: bit di parità intercalati

(e) RAID 4: blocchi di parità intercalati

(f) RAID 5: blocchi intercalati a parità distribuita

(g) RAID 6: ridondanza P + Q

Figura 12.11 Livelli RAID.

graficamente (nella figura, la lettera *P* indica i bit di correzione degli errori, la lettera *C* indica una seconda copia dei dati). In tutti i casi riportati nella figura, sono presenti quattro dischi di dati, mentre i dischi supplementari s'impiegano per memorizzare le informazioni ridondanti per il ripristino dai guasti.

- RAID **di livello 0.** Il livello 0 si riferisce a batterie di dischi con sezionamento a livello dei blocchi, ma senza ridondanza (come la copiatura speculare o i bit di parità), come illustrato nella Figura 12.11(a).
- RAID **di livello 1.** Il livello 1 si riferisce alla tecnica della copiatura speculare. La Figura 12.11(b) mostra un'organizzazione basata sulla copiatura speculare.
- RAID **di livello 2.** Il livello 2 è anche noto come **organizzazione con codici per la correzione degli errori** (*error-correcting codes*, ECC). Da molto tempo i sistemi di memorizzazione impiegano tecniche di riconoscimento degli errori basate sui bit di parità. In un sistema di questo tipo, ogni byte di memoria ha associato un bit di parità che indica se i bit con valore 1 nel byte sono in numero pari (parità = 0) oppure dispari (parità = 1). Se si altera uno dei bit nel byte (un valore 1 diventa 0 o viceversa), la parità del byte cambia e quindi non concorda più con la parità memorizzata. Analogamente, se si altera il bit di parità, esso non concorda più con la parità calcolata. In questo modo s'identificano tutti gli errori di un singolo bit nel sistema di memoria. Gli schemi di correzione degli errori memorizzano due o più bit supplementari e possono ricostruire i dati nel caso di un singolo bit danneggiato. La stessa idea alla base dell'ECC si può usare immediatamente nelle batterie di dischi eseguendo il sezionamento dei byte presenti nei dischi. Ad esempio, il primo bit di ogni byte si potrebbe memorizzare nel disco 1, il secondo bit nel disco 2, e così via fino alla memorizzazione dell'ottavo bit nel disco 8 e alla memorizzazione dei bit di correzione degli errori in ulteriori dischi. Questo schema è rappresentato graficamente nella Figura 12.11(c), dove i dischi etichettati con la lettera *P* contengono i bit di correzione. Se uno dei dischi si guasta, i bit rimanenti del byte e i bit di correzione a esso associati si possono leggere dagli altri dischi e usare per ricostruire i dati danneggiati. Si noti che il RAID di livello 2 richiede tre soli dischi in più per quattro dischi di dati, a differenza del RAID di livello 1, che richiede quattro dischi in più.
- RAID **di livello 3.** Con il livello 3, o **organizzazione con bit di parità intercalati**, si migliora l'organizzazione del livello 2 considerando che, a differenza dei sistemi di memoria centrale, i controllori dei dischi possono rilevare se un settore è stato letto correttamente, così che un unico bit di parità si può usare sia per individuare gli errori sia per correggerli. L'idea è la seguente: se uno dei settori è danneggiato, si conosce esattamente di quale settore si tratta e, per ogni bit nel settore, è possibile determinare se debba avere valore 1 o 0 calcolando la parità dei bit corrispondenti dai settori negli altri dischi. Se la parità dei rimanenti bit è uguale a quella memorizzata, il bit mancante è 0, altrimenti è 1. Il RAID di livello 3 è altrettanto valido del livello 2 ma richiede un solo disco supplementare, quindi il RAID di livello 2 non è utilizzato nella pratica. La Figura 12.11(d) illustra questo schema.

  Il livello 3 presenta due vantaggi rispetto a livello 1. Il primo vantaggio è che si usa un solo disco per la parità dei dati memorizzati in diversi dischi di dati, anziché un ulteriore disco per ciascun disco di dati come nel livello 1. Il secondo vantaggio è che, essendo le letture e le scritture dei byte distribuite su più dischi con un sezionamento dei dati a *n* vie, la velocità di trasferimento di un singolo blocco è pari a *n* volte quella dei RAID di livello 1. D'altro canto, però, il livello 3 permette meno operazioni di I/O al secondo, perché ogni disco è coinvolto da tutte le richieste.

Un altro problema di prestazioni riguardante il RAID di livello 3 (come per tutti i livelli RAID basati sui bit di parità) è il tempo richiesto dal calcolo e dalla scrittura della parità. Questo tempo aggiuntivo determina operazioni di scrittura significativamente più lente rispetto a batterie RAID senza parità. Per limitare questo calo di prestazioni, molte batterie RAID dispongono di un controllore capace di gestire il calcolo della parità. Questo sposta il carico dovuto al calcolo della parità dalla CPU alla batteria di dischi. La batteria ha anche una cache RAM **non volatile** (NVRAM) per memorizzare i blocchi mentre viene calcolata la parità e per memorizzare transitoriamente le scritture dal controllore ai dischi. Questa combinazione può rendere la tecnica RAID con parità altrettanto veloce di quella senza parità; infatti, una batteria RAID con cache e bit di parità può avere prestazioni migliori di un'organizzazione RAID senza cache e senza parità.

- RAID **di livello 4.** Nel livello 4, o **organizzazione con blocchi di parità intercalati**, s'impiega il sezionamento a livello dei blocchi, come nel RAID di livello 0 e inoltre si tiene un blocco di parità in un disco separato per i blocchi corrispondenti presenti in n dischi diversi da questo. Tale schema è illustrato nella Figura 12.11(e). Se uno dei dischi si guasta, il blocco di parità si può usare insieme ai blocchi corrispondenti degli altri dischi per ripristinare i blocchi nel disco guasto.

  La lettura di un blocco richiede l'accesso a un solo disco, permettendo la gestione di altre richieste da parte di altri dischi. Quindi, la capacità di trasferimento dei dati per ciascun accesso è minore, ma gli accessi per la lettura possono procedere in modo parallelo ottenendo una rapidità complessiva nell'I/O più alta. La capacità di trasferimento per la lettura di molti dati è alta, poiché si possono leggere in modo parallelo tutti i dischi e anche le operazioni di scrittura di grandi quantità di dati presentano un'alta capacità di trasferimento, poiché i dati e i bit di parità si possono scrivere in parallelo.

  Scritture indipendenti di modesta entità non si possono eseguire in parallelo. La scrittura di dati pari a meno di un blocco richiede la lettura del blocco, la sua modifica, e la scrittura del blocco modificato; anche il blocco di parità deve essere aggiornato. Si parla a questo proposito del **ciclo lettura-modifica-scrittura**. Una singola richiesta di scrittura comporta pertanto quattro accessi al disco, due in lettura e due in scrittura.

  Nel Capitolo 11 è stato presentato WAFL: esso adotta RAID di livello 4, che permette l'aggiunta di dischi al sistema senza soluzione di continuità. Inizializzando i nuovi dischi a zero, la parità non cambia, e la batteria RAID è ancora nello stato corretto.

- RAID **di livello 5.** Il livello 5, o **organizzazione con blocchi intercalati a parità distribuita**, differisce dal livello 4 per il fatto che, invece di memorizzare i dati in n dischi e la parità in un disco separato, i dati e le informazioni di parità sono distribuite tra gli $n + 1$ dischi. Per ogni blocco, uno dei dischi memorizza la parità e gli altri i dati. Ad esempio, considerando una batteria di cinque dischi, la parità per il blocco m-esimo si memorizza nel disco (m mod 5) + 1, mentre i blocchi m-esimi degli altri quattro dischi contengono i dati effettivi per quel blocco. Questo schema è illustrato nella Figura 12.11(f), dove i simboli P sono distribuiti su tutti i dischi. Un blocco di parità non può contenere informazioni di parità per blocchi che risiedono nello stesso disco, poiché un guasto al disco provocherebbe sia la perdita di dati sia la perdita dell'informazione di parità e quindi i dati non sarebbero ripristinabili. Con la distribuzione della parità sui diversi dischi, il RAID di livello 5 evita un uso intensivo del disco dove risiede la parità, che invece si ha con il RAID di livello 4. RAID 5 è il più comune sistema di parità RAID.

- **RAID di livello 6.** Il livello 6, detto anche **schema di ridondanza P + Q**, è molto simile al RAID di livello 5, ma memorizza ulteriori informazioni ridondanti per poter gestire guasti contemporanei di più dischi. Invece di usare la parità, s'impiegano codici per la correzione degli errori come i **codici di Reed-Solomon**. Nello schema mostrato nella Figura 12.11(g), sono memorizzati 2 bit di dati ridondanti ogni 4 bit di dati effettivi (a differenza di 1 bit di parità usato nel livello 5) e il sistema risultante può tollerare due guasti dei dischi.
- **RAID di livello 0 + 1.** Il livello 0 + 1 consiste in una combinazione dei livelli RAID 0 e 1. Il livello 0 fornisce le prestazioni, mentre il livello 1 l'affidabilità. Di solito, questo schema porta a prestazioni migliori rispetto a livello 5 e si usa prevalentemente negli ambienti in cui sono importanti sia le prestazioni sia l'affidabilità. Sfortunatamente, questo schema richiede, come il RAID di livello 1, un raddoppio del numero di dischi necessario per memorizzare i dati, quindi è anche più costoso del RAID di livello 5. Nel RAID di livello 0 + 1, si sezionano i dati presenti in un insieme di dischi e si duplica ogni sezione con la tecnica della copiatura speculare.

  Un altro metodo che sta diventando disponibile commercialmente è il RAID di livello 1 + 0, in cui si fa prima la copiatura speculare dei dischi a coppie, e poi il sezionamento su queste coppie. Questo schema RAID ha alcuni vantaggi teorici rispetto al RAID 0 + 1. Ad esempio, se si guasta un singolo disco nel RAID 0 + 1, l'intera sezione di dati diventa inaccessibile, lasciando disponibile solo l'altra sezione. Con un guasto nel RAID 1 + 0, il singolo disco diventa inaccessibile, ma il suo duplicato è ancora disponibile, come tutti gli altri dischi (Figura 12.12).

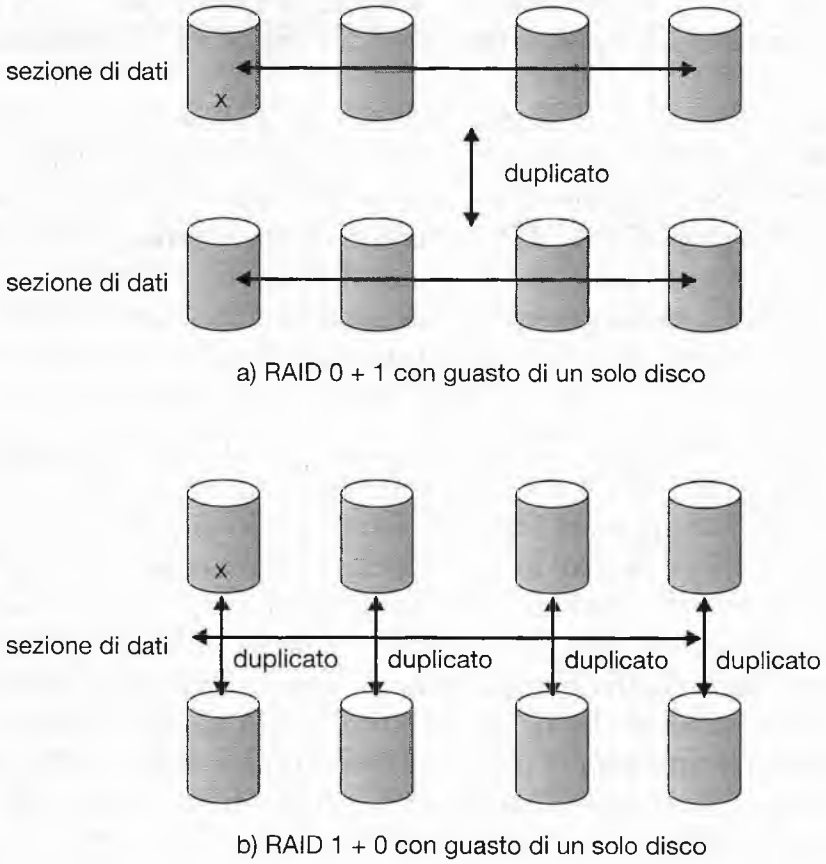


Figura 12.12 RAID 0 + 1 e 1 + 0.

Infine, si deve considerare che sono state proposte numerose altre varianti agli schemi RAID di base illustrati sopra e questo ha portato anche una certa confusione nelle precise definizioni dei diversi livelli RAID.

Un altro punto soggetto a molte varianti è l'implementazione del RAID. Esaminiamo i diversi livelli a cui è possibile implementare un sistema RAID.

- Il software per la gestione dei volumi può implementare un sistema RAID all'interno del kernel o a livello dei programmi di sistema. In questo caso, nonostante i dispositivi per la memorizzazione possano fornire funzionalità minime, è possibile ottenere un sistema RAID completo. Il metodo RAID con parità è alquanto lento se realizzato tramite software, quindi gli si preferisce solitamente RAID 0,1 oppure 0 + 1.
- Il metodo RAID può essere implementato a livello hardware dall'adattatore del bus della macchina (*host bus adapter*, HBA). Solo i dischi connessi direttamente all'HBA possono costituire parte integrante di una data batteria RAID. Questa soluzione è a basso costo, ma non può dirsi molto flessibile.
- Il metodo RAID può essere implementato a livello hardware dalla batteria di dischi. È così possibile creare sistemi RAID a vari livelli, e persino ricavare da essi volumi più piccoli, che sono quindi presentati al sistema operativo, che avrà solo da realizzare il file system su ciascuno dei volumi. Le batterie possono disporre di connessioni multiple o far parte di una rete di memorizzazione secondaria (SAN), consentendo a vari terminali di sfruttare le funzionalità della batteria.
- Il metodo RAID può essere implementato dai dispositivi di virtualizzazione del disco a livello di interconnessione SAN. In questo caso, un dispositivo funge da intermediario tra i terminali e l'area di memorizzazione, accettando istruzioni dai server e gestendo l'accesso alla memoria secondaria. Esso potrebbe, ad esempio, attuare la copiatura speculare, trascrivendo ciascun blocco su due dispositivi distinti per la memorizzazione.

Ulteriori funzionalità, come quella di istantanea e di replica, possono essere implementate a ognuno di questi livelli. La **replica**, che può essere sincrona o asincrona, concerne la duplicazione automatica di scritture su siti diversi, per finalità di ridondanza, o di ripristino in caso di danneggiamenti. Se è sincrona, ciascun blocco deve essere scritto sia localmente, sia in remoto, prima che la scrittura sia considerata completa; se è asincrona, si effettuano scritture per gruppi e a cicli periodici. La replica asincrona espone al rischio di perdere i dati, se il sito principale fallisce, ma è più veloce e non ha limiti di distanza.

L'implementazione di queste funzionalità varia a seconda del livello scelto per realizzare il sistema RAID. Qualora RAID, per esempio, sia implementato a livello software, ciascuna macchina può aver necessità di implementare e gestire la replica per proprio conto. Tuttavia, se la replica avviene a livello della batteria di dischi o dell'interconnessione SAN, si possono replicare i dati della macchina a prescindere dalla piattaforma del sistema operativo e dalle sue funzionalità.

Un'altra caratteristica spesso presente nei sistemi RAID è la previsione di **dischi di scorta** (*hot spare*), che possono sostituire quelli normali. Quando, infatti, un disco presenta difetti di funzionamento, subentra al suo posto un disco di scorta, ad esempio per ricostruire un disco danneggiato, che appartiene a una coppia configurata per la copiatura speculare. In questo modo, si può ristabilire automaticamente lo stato corretto del livello RAID, senza attendere che il disco difettoso sia sostituito. È possibile riparare più di un guasto, senza l'intervento di un operatore, con l'allocazione di più dischi di scorta.

12.7.4 Scelta di un livello RAID

Viste le svariate possibilità esistenti, ci si potrebbe chiedere quali siano i criteri di scelta dei progettisti nei confronti del livello RAID. Se un disco si guasta, il tempo necessario a ricostruire i dati che contiene può essere rilevante. Questo fattore può essere importante nel caso in cui venga richiesto un flusso continuo di dati, come nei sistemi ad alte prestazioni o nei sistemi interattivi di basi di dati. Inoltre, le prestazioni del processo di ricostruzione influenzano il tempo medio di guasto.

Queste prestazioni possono variare a seconda del livello RAID utilizzato. La ricostruzione più semplice si ha per RAID di livello 1, poiché i dati possono essere copiati da un altro disco; per gli altri livelli, per ricostruire i dati in un disco guasto è necessario accedere a tutti gli altri dischi della batteria. Il tempo necessario per la ricostruzione dei dati può variare nell'ordine di ore nel caso di sistemi RAID di livello 5 con molti dischi.

RAID di livello 0 si usa nelle applicazioni ad alte prestazioni in cui le perdite di dati non sono critiche. Il RAID di livello 1 si usa comunemente nelle applicazioni che richiedono un'alta affidabilità e un rapido ripristino. I livelli RAID 0 + 1 e 1 + 0 si usano dove le prestazioni e l'affidabilità sono importanti, ad esempio per piccole basi di dati. A causa dell'elevata richiesta di spazio del RAID di livello 1, per la memorizzazione di grandi quantità di dati, spesso si preferisce impiegare il RAID di livello 5. Il livello 6, attualmente non disponibile in molti sistemi RAID, dovrebbe offrire una migliore affidabilità rispetto a livello 5.

Progettisti e amministratori di sistemi RAID devono prendere anche altre decisioni importanti, ad esempio riguardo al numero ottimale di dischi in una batteria e al numero di bit che ciascun bit di parità deve proteggere. Maggiore è il numero di dischi in una batteria, maggiore sarà la capacità di trasferimento dei dati, ma il sistema sarà anche più costoso. Maggiore è il numero di bit protetti da un singolo bit di parità, minore sarà lo spazio richiesto dai bit di parità; sarà però maggiore anche la probabilità che un secondo disco si guasti prima che un disco guasto sia riparato e questo porterebbe alla perdita di dati.

12.7.5 Estensioni

I concetti relativi ai sistemi RAID sono stati generalizzati ad altri dispositivi di memorizzazione, comprese le batterie di nastri e anche alla diffusione dei dati tramite sistemi senza fili (*wireless*). Con strutture RAID applicate alle batterie di nastri si possono ripristinare i dati anche se uno dei nastri della batteria è danneggiato. Se applicate alla trasmissione dei dati, si divide ogni blocco di dati in unità più piccole che si trasmettono insieme a un'unità di parità; se per qualsiasi ragione una delle unità non viene ricevuta, può essere ricostruita dalle altre. Di solito, con l'uso di unità automatiche dotate di molte unità a nastro si esegue il sezionamento dei dati su tutte le unità per aumentare la produttività e diminuire il tempo di trasferimento dei dati.

12.7.6 Problemi connessi a RAID

I sistemi RAID, purtroppo, non assicurano sempre la disponibilità dei dati al sistema operativo e ai loro utenti. Un puntatore potrebbe indicare il file sbagliato, per esempio, e lo stesso potrebbe accadere ai puntatori nella struttura interna dei file. Le operazioni incomplete di scrittura, se non ripristinate in maniera adeguata, possono alterare i dati. Altri processi, inoltre, potrebbero scrivere accidentalmente sulle strutture del file system. Il metodo RAID protegge dagli errori derivanti dai supporti fisici per la memorizzazione, ma non da altri tipi di errori dovuti ai dispositivi e ai programmi. I pericoli potenziali, per i dati di un sistema, si estendono alla totalità degli errori derivanti dal software e dall'hardware.

LA BATTERIA InServ

Il progresso, grazie al quale sono di continuo introdotte soluzioni migliori, più veloci e meno costose, ridefinisce spesso i confini che separano le tecnologie già esistenti. Si consideri, per esempio, la batteria InServ di 3Par. A differenza di molte altre, essa non richiede la configurazione di un insieme di dischi a un livello RAID specifico, ma scompone invece ogni disco in porzioni da 256 MB. Il metodo RAID è pertanto applicato a livello di queste porzioni. Di conseguenza, vari livelli RAID possono interessare lo stesso disco, visto che le porzioni servono per formare volumi multipli.

La batteria InServ mette inoltre a disposizione le istantanee, una funzionalità simile a quella del file system WAFL. Le istantanee di InServ prevedono sia il formato lettura-scrittura sia il formato a sola lettura, consentendo a utenti multipli di montare copie di un dato file system senza doverlo possedere integralmente. Le modifiche eventualmente apportate dagli utenti alla propria copia sono di copiatura su scrittura, ragion per cui non hanno effetto sulle altre copie.

Un'altra innovazione è il cosiddetto **utility storage**. Alcuni file system non possono essere espansi, né compressi. I file system di questo genere mantengono costantemente le dimensioni originali: per qualsiasi modifica è necessaria la copiatura di dati. Un amministratore può configurare InServ per fornire a una macchina cospicue quantità di memoria logica, che all'inizio occupano solamente un piccolo spazio di memoria fisica. Mentre la macchina comincia a usare lo spazio di memorizzazione, dischi non ancora utilizzati sono assegnati alla macchina, fino a raggiungere il livello logico originale. In tal modo, la macchina utente è indotta a credere di possedere un vasto spazio di memorizzazione permanente, dove creare i propri file system, e così via. InServ può aggiungere o rimuovere dischi dal file system, senza che se ne accorga. Questa caratteristica può ridurre il numero di unità a disco necessarie agli utenti, o perlomeno ritardare l'acquisizione di nuovi dischi finché non divengano realmente necessari.

Per risolvere tali problemi, il file system Solaris ZFS ricorre a una strategia innovativa. Esso applica una somma di controllo (*checksum*) interna a ogni blocco, dati e metadati inclusi. Un'ulteriore funzionalità deriva dalla collocazione delle somme di controllo, che non risiedono nel blocco sottoposto a controllo: ciascuna di loro, invece, è memorizzata insieme al puntatore a quel blocco (Figura 12.13). Si consideri un inode con puntatori ai propri dati. All'interno dell'inode si trova la somma di controllo per ciascun blocco di dati. Se si verifica

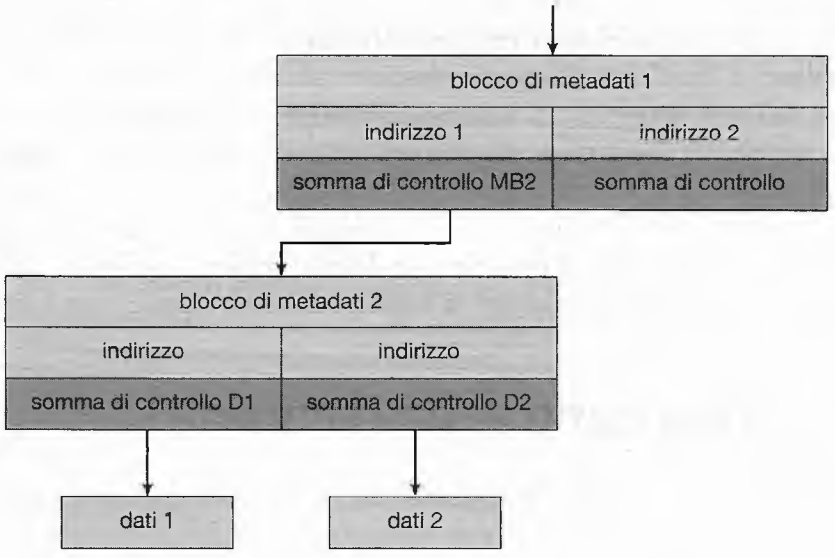


Figura 12.13 ZFS applica una somma di controllo a tutti i metadati e i dati.

un problema con i dati, la somma di controllo darà un valore errato e il file system verrà a conoscenza del problema. Qualora sia attiva la copiatura speculare, il sistema ZFS, in presenza di un blocco con una somma di controllo corretta e di uno con una somma errata, sostituirà automaticamente il blocco errato con quello valido. In maniera simile, l'elemento della directory che punta all'inode possiede una somma di controllo relativa all'inode. Qualunque problema riguardi l'inode, quindi, è rilevato dall'accesso alla directory. Queste somme di controllo, che sono applicate a tutte le strutture di ZFS, producono risultati molto più efficaci degli ambienti RAID o dei file system tradizionali, per livello di coerenza, rilevazione degli errori e capacità di correggerli. Il sovraccarico di gestione determinato dalle somme di controllo e dai cicli supplementari di lettura-modifica-scrittura dei blocchi non condizionano il funzionamento complessivo di ZFS, che mantiene un'alta velocità nelle prestazioni.

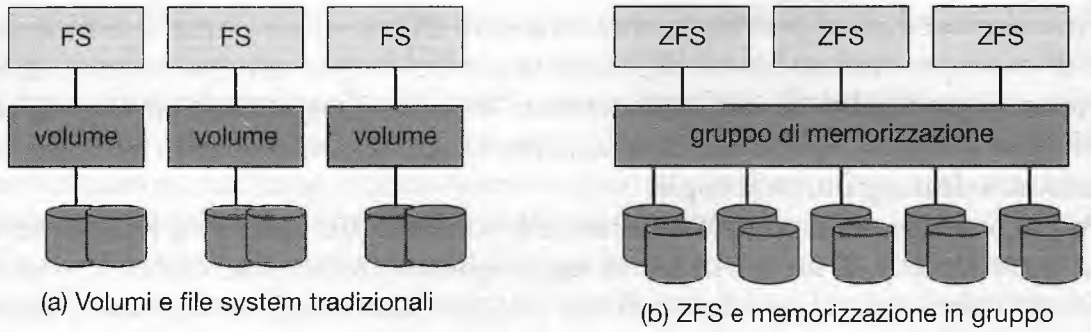
La maggior parte delle implementazioni RAID è caratterizzata dalla mancanza di flessibilità. Considerate una batteria di memorizzazione dotata di venti dischi divisi in quattro insiemi da cinque dischi. Ogni gruppo di cinque dischi è un insieme RAID di livello 5. Ne risultano quattro volumi separati, ciascuno contenente un proprio file system. Ma che cosa succede se un file system ha una dimensione eccessiva per un insieme RAID di livello 5 a cinque dischi? E se un altro file system necessita di un'area ridottissima? Se tali fattori sono noti in anticipo, allora i dischi e i volumi possono essere allocati adeguatamente. L'utilizzo del disco e le richieste variano però molto frequentemente nel tempo.

Anche se la batteria di memorizzazione ha permesso all'intero insieme di venti dischi di essere creato come un grande insieme RAID, potrebbero insorgere altre problematiche. Nell'insieme potrebbero essere costruiti diversi volumi di dimensioni differenti. Alcuni gestori del volume non ci permettono però di cambiare la dimensione di un volume. In quel caso si ripresenterebbe la stessa situazione descritta in precedenza, ovvero dimensioni discrepanti di file system. Alcuni gestori di volume permettono cambiamenti di dimensione, ma alcuni file system non permettono l'aumento o la diminuzione della dimensione del file system stesso. I volumi potrebbero cambiare dimensione, ma i file system dovrebbero essere ricreati per poter usufruire di quei cambiamenti.

ZFS combina la gestione dei file system e quella dei volumi in una unità in grado di offrire una maggiore funzionalità rispetto a quella permessa dalla tradizionale separazione di tali funzioni. I dischi, o le partizioni di dischi, sono riuniti in **gruppi di memorizzazione** (*pools of storage*) attraverso insiemi RAID. Un gruppo può contenere uno o più file system ZFS. Tutta l'area libera di un gruppo è a disposizione di tutti i file system contenuti all'interno di quel gruppo. Quando i blocchi vengono utilizzati e liberati all'interno del file system ZFS utilizza "malloc" e "free" per allocare e rilasciare memoria in ogni file system. Di conseguenza non ci sono limiti artificiali all'utilizzo della memoria e non sussiste la necessità di ridistribuire i file system tra i volumi né di ridimensionare i volumi. ZFS stabilisce delle quote per limitare la dimensione di un file system e definisce dei parametri per assicurarsi che il file system possa aumentare all'interno di una dimensione specificata, ma queste variabili possono essere sempre cambiate dal proprietario del file system. La Figura 12.14(a) illustra i volumi e i file system tradizionali, mentre la Figura 12.14(b) rappresenta il modello ZFS.

12.8 Realizzazione della memoria stabile

Nel Capitolo 6 è stata presentata la registrazione con scrittura anticipata (*write-ahead logging*), che richiede la disponibilità di una memoria stabile. Per definizione, le informazioni residenti in questo tipo di memoria non vanno *mai* perse. Per realizzare un tale tipo di



(a) Volumi e file system tradizionali

(b) ZFS e memorizzazione in gruppo

Figura 12.14 (a) Volumi e file system tradizionali. (b) Gruppo e file system ZFS.

memoria si devono replicare le informazioni necessarie in più dispositivi di memorizzazione (di solito dischi) con modalità di guasto indipendenti. Inoltre è necessario coordinare l'aggiornamento delle informazioni in modo tale che un eventuale malfunzionamento durante l'aggiornamento non lasci tutte le copie in uno stato danneggiato, e che durante il ripristino delle informazioni a seguito di un guasto sia possibile riportare ogni copia alla sua forma corretta anche se si verifica un altro malfunzionamento proprio durante il ripristino. Nel resto del paragrafo si presentano le possibili soluzioni a queste esigenze.

Un'operazione di scrittura in un disco può avere uno dei seguenti esiti.

1. **Operazione riuscita.** I dati sono stati scritti correttamente nel disco.
2. **Insuccesso parziale.** Si è verificato un guasto durante il trasferimento, e solo alcuni tra i settori coinvolti sono stati correttamente aggiornati, mentre il settore interessato dalla scrittura al momento del malfunzionamento può essere stato danneggiato.
3. **Insuccesso totale.** Il malfunzionamento è avvenuto prima dell'avvio del processo di scrittura nel disco; i dati già residenti nel disco sono rimasti inalterati.

Si richiede che il sistema riconosca un guasto verificatosi durante il trasferimento e invochi una procedura di ripristino per riportare il blocco in uno stato coerente. A tale scopo il sistema deve mantenere due blocchi fisici per ciascun blocco logico, eseguendo ogni operazione nel modo seguente:

1. scrittura delle informazioni nel primo blocco fisico;
2. completata con successo la prima scrittura, scrittura delle stesse informazioni nel secondo blocco fisico;
3. l'operazione è considerata completa solamente dopo che la seconda scrittura è stata eseguita correttamente.

Durante il ripristino dovuto a un malfunzionamento si esamina ogni coppia di blocchi fisici; se essi contengono gli stessi dati e non c'è traccia d'errori, non è necessario intraprendere alcuna azione ulteriore. Se un blocco contiene un errore riscontrabile, se ne sostituisce il contenuto con quello del secondo blocco. Se in nessuno dei due blocchi si riscontra un errore, ma ciascuno contiene informazioni differenti, si sostituisce il contenuto del primo blocco con quello del secondo. Questa procedura di ripristino assicura che gli unici due esiti possibili di un'operazione di scrittura nella memoria stabile siano o la completa riuscita dell'operazione oppure il suo insuccesso totale, in quest'ultimo caso i dati memorizzati non subiscono alcun cambiamento.

Questa procedura si può facilmente estendere all'uso di un numero arbitrariamente grande di copie per ciascun blocco di memoria stabile. Benché un gran numero di queste copie riduca la probabilità di malfunzionamenti, di solito è ragionevole simulare la memoria stabile con due sole copie. I dati di una memoria stabile non andranno mai persi purché un guasto non distrugga tutte le copie.

Poiché le attese per il completamento delle scritture (I/O sincrono) richiedono molto tempo, molte batterie di memorizzazione aggiungono NVRAM come cache. Poiché la memoria è non volatile (di solito è dotata di una pila per l'alimentazione elettrica di riserva), si può ritenere affidabile come un disco per la memorizzazione dei dati, e si può considerare parte della memoria stabile. Le scritture in essa sono molto più rapide di quelle nei dischi, quindi le prestazioni migliorano notevolmente.

12.9 Strutture per la memorizzazione terziaria

Nessuno comprerebbe un videoregistratore contenente un'unica cassetta non sostituibile o non estraibile, o un lettore di CD o DVD con un disco bloccato all'interno: ci si attende di poter usare un videoregistratore o un lettore di CD con videocassette o dischi diversi e relativamente poco costosi. Anche nei calcolatori l'uso di supporti di memorizzazione rimovibili economici e di un solo dispositivo di lettura o scrittura riduce i costi complessivi. Il costo contenuto è la caratteristica distintiva della memoria terziaria, argometo di questo paragrafo.

12.9.1 Dispositivi per la memorizzazione terziaria

La memoria terziaria consiste di **mezzi rimovibili**, di cui dischetti, CD-ROM e DVD sono gli esempi più comuni. Sul mercato sono disponibili anche molti altri prodotti, fra i quali dispositivi rimovibili che interagiscono con il computer tramite un'interfaccia USB, basati su memorie flash.

12.9.1.1 Dischi rimovibili

I dischi rimovibili sono un tipo di memoria terziaria. I dischetti sono un esempio di dischi magnetici rimovibili, costituiti da un disco sottile e flessibile, ricoperto di materiale magnetico e racchiuso in un involucro protettivo di plastica. I comuni dischetti hanno una capacità di circa 1 MB, ma una tecnologia simile si usa per costruire dischi magnetici rimovibili della capacità di più di 1 GB. I dischi magnetici rimovibili possono funzionare a una velocità quasi pari a quella di un'unità a disco, anche se il rischio che la loro superficie sia danneggiata da graffi è maggiore.

I **dischi magneto-ottici** sono un altro tipo di dischi rimovibili: anche in questo caso i dati sono registrati su un disco ricoperto di materiale magnetico, ma la tecnologia impiegata per la registrazione è molto diversa da quella dei dischi magnetici. La testina di un disco magneto-ottico è sospesa a una distanza dalla superficie del disco molto maggiore rispetto alla testina di un disco magnetico, e il materiale magnetico è protetto da uno spesso strato di plastica o di vetro; di conseguenza, il disco è molto più resistente a eventuali collisioni della testina.

L'unità per i dischi magneto-ottici ha una bobina capace di produrre un campo magnetico, ma a temperature ordinarie il campo è troppo diffuso e debole per poter magnetizzare un bit sul disco; per rimediare a questo fatto la testina emette un raggio laser verso la superficie del disco, puntandolo sulla piccolissima area dove si vuole scrivere un bit. Il surriscaldamento dell'area provocato dal laser fa sì che essa divenga sensibile al campo magnetico, ed è in questo modo che un campo magnetico debole e diffuso riesce a registrare un minuscolo bit.

La testina di un disco magneto-ottico è troppo lontana dalla superficie del disco per poter leggere i dati rilevando i piccolissimi campi magnetici in modo analogo a quanto fa la testina di un disco magnetico. Perciò l'unità a disco legge i bit sfruttando una proprietà della luce laser detta **effetto Kerr**; quando un raggio laser è riflesso da un punto magnetizzato la sua polarizzazione è ruotata in senso orario o antiorario secondo l'orientazione del campo magnetico: per leggere i bit la testina rileva questa rotazione.

Un'altra categoria di dischi rimovibili è quella dei **dischi ottici**, i quali non sfruttano per niente il magnetismo, ma usano materiali speciali che la luce laser può alterare in modo da creare punti relativamente chiari o scuri. Un esempio di tecnologia per dischi ottici sono i **dischi a cambio di fase**, ricoperti da un materiale che può solidificare passando a uno stato cristallino o a uno stato amorfo. Lo stato cristallino è più trasparente, perciò un raggio laser è più luminoso quando attraversa il materiale a cambio di fase ed è riflesso dall'apposito strato. Le unità per dischi a cambio di fase impiegano laser capaci di emettere raggi a tre differenti livelli di potenza: bassa, per leggere i dati; media, per cancellare il disco fondendo e facendo solidificare il supporto di memorizzazione nello stato cristallino; alta, per scrivere nel disco fondendo e facendo solidificare il supporto nello stato amorfo. Gli esempi più comuni di questo tipo di tecnologia sono i dischi ottici riscrivibili CD-RW e DVD-RW.

Il tipo di dischi fin qui descritti si possono riutilizzare: per questo sono detti **dischi a lettura e scrittura**. Per contro, i **dischi monoscrivibili** o (*write once, read many times*, WORM) costituiscono una categoria distinta. Un vecchio modo di costruire un disco WORM è di inserire una pellicola d'alluminio tra due piatti di plastica o di vetro. Per scrivere un bit l'unità usa un raggio laser per praticare un piccolo foro nell'alluminio: poiché questo processo non è reversibile, si può scrivere una sola volta su un qualunque settore del disco. Sebbene sia possibile distruggere l'informazione contenuta in un disco WORM, ad esempio praticando fori dappertutto, è praticamente impossibile alterare i dati in esso contenuti, perché l'unica azione possibile è quella di aggiungere fori, ed è assai probabile che il codice ECC associato a ogni settore rilevi le modifiche. I dischi WORM sono considerati durevoli e affidabili: lo strato di metallo è protetto dalla copertura di vetro o plastica, e i campi magnetici non possono danneggiare la registrazione. Una più recente tecnologia di monoscrittura compie la registrazione su un pigmento polimerico invece che su uno strato d'alluminio: il pigmento forma dei punti assorbendo la luce del laser. Questa tecnologia s'impiega nei dischi ottici scrivibili CD-R e DVD-R.

I **dischi a sola lettura**, ad esempio i CD-ROM e i DVD, sono commercializzati con un contenuto preregistrato: fanno uso di una tecnologia simile a quella dei dischi WORM (sebbene in questo caso i fori siano stampati) e sono assai durevoli.

Generalmente i dischi rimovibili sono più lenti dei corrispondenti dischi fissi. Il processo di scrittura è più lento così come la rotazione e talvolta la ricerca.

12.9.1.2 Nastri

I nastri magnetici sono un altro tipo di memorizzazione terziaria. In linea generale un nastro può contenere più dati di un disco ottico o magnetico rimovibile. Le unità a nastro e quelle a disco hanno velocità di trasferimento simili ma, poiché richiedono operazioni di avanzamento rapido o di riavvolgimento che possono durare decine di secondi o addirittura interi minuti, l'accesso diretto è molto più lento per un nastro che per un disco.

Anche se un'unità a nastro è più costosa di un'unità a disco, una cartuccia a nastro è più economica di un disco magnetico della stessa capacità: i nastri magnetici sono quindi un mezzo conveniente qualora non si richieda la possibilità di rapidi accessi diretti. I nastri si usano comunemente per contenere copie di riserva dei dati presenti nei dischi, ma si usano

anche nei grandi centri di calcolo, dotati di supercalcolatori, per memorizzare le enormi quantità di dati che s'impiegano nella ricerca scientifica o per la gestione di grandi aziende.

Grandi stazioni di registrazione a nastri usano tipicamente meccanismi automatici per spostare i nastri dalle unità ad appositi contenitori in un archivio di nastri; questi meccanismi offrono ai calcolatori l'accesso automatico a un elevato numero di nastri.

Un archivio automatizzato riduce i costi totali della registrazione dei dati. Un file non immediatamente necessario residente in un disco si può **archiviare** in un nastro a un costo per gigabyte che può essere inferiore; quando il file si renderà necessario, il calcolatore potrà installarlo nuovamente nel disco. Un archivio automatizzato realizza un tipo di memorizzazione talvolta detto **quasi in linea**, perché si situa fra le alte prestazioni della memorizzazione **in linea** nei dischi magnetici e il basso costo di una memorizzazione **non in linea** in nastri archiviati in qualche deposito.

12.9.1.3 Tecnologie future

È possibile che in futuro altre tecnologie di memorizzazione acquisiscano importanza. A volte sono le vecchie tecnologie a essere utilizzate in nuovi modi, parallelamente ai cambiamenti economici e all'evoluzione tecnologica. Ad esempio, i **dischi a stato solido**, o SSD, stanno diventando sempre più importanti e diffusi. In breve, un SSD è un disco utilizzato come un disco rigido che, a seconda della tecnologia di memorizzazione utilizzata, può essere volatile o non volatile. La tecnologia di memorizzazione incide inoltre sulle prestazioni. I dischi a stato solido non volatili presentano le stesse caratteristiche dei dischi rigidi tradizionali, ma possono essere più affidabili, perché non hanno parti mobili, e più veloci, perché non hanno tempo di ricerca né di latenza. Richiedono inoltre meno energia. Rispetto alle unità a disco tradizionali i dischi a stato solido hanno però un costo al megabyte più elevato, una minore capacità dei dischi rigidi più grandi e una durata di vita a volte più breve; per questi motivi, il loro utilizzo è limitato. Per fare un esempio, i dischi a stato solido sono impiegati nelle batterie di memorizzazione per contenere metadati che richiedono prestazioni elevate, come il log di un file system con annotazioni delle modifiche. I dischi a stato solido vengono anche montati sui computer portatili per renderli più leggeri, più veloci e più efficienti in termini energetici.

Un'altra promettente tecnologia per la memorizzazione di massa è la **memoria olografica**, che sfrutta la luce laser per registrare fotografie olografiche su supporti dedicati.

Un **ologramma** può essere pensato come una matrice tridimensionale di pixel da 1 bit ciascuno, i cui valori 0 e 1 rappresentano il nero e il bianco, rispettivamente. Poiché è possibile trasferire tutti i pixel tramite un solo fascio di luce laser, la velocità di trasmissione è altissima. In futuro, grazie ai progressi della ricerca, la memoria olografica potrebbe diventare una concreta opzione commerciale.

Un'altra tecnologia di memorizzazione oggetto di attive ricerche si fonda sui **sistemi meccanici microelettronici** (*microeletronic mechanical systems*, MEMS). L'idea consiste nell'applicare le tecnologie che s'impiegano nella fabbricazione dei circuiti integrati per costruire piccole macchine di memorizzazione. Una proposta prevede la fabbricazione di una matrice di 10.000 minuscole testine di disco, con un centimetro quadrato di materiale magnetico di registrazione sospeso sopra tale matrice. Quando si muove il materiale di registrazione lungo le testine, ognuna di loro accede alla propria traccia lineare di dati presente sul materiale magnetico. Il materiale di registrazione si può traslare leggermente in modo laterale per consentire a tutte le testine di accedere alla loro traccia successiva. Sebbene resti da vedere se questa tecnologia possa avere successo, essa può offrire sistemi di memorizzazione non volatile più rapidi dei dischi magnetici e più economici delle memorie a semiconduttori DRAM.

Indipendentemente dal fatto che il supporto di memorizzazione in uso sia un disco magnetico rimovibile, un DVD, o un nastro magnetico, il sistema operativo deve offrire una serie di funzioni affinché i mezzi rimovibili siano utilizzabili per contenere dati: questo argomento è trattato nel Paragrafo 12.9.2.

12.9.2 Compiti del sistema operativo

Due tra gli obiettivi primari di un sistema operativo sono la gestione dei dispositivi fisici e la presentazione di una macchina virtuale alle applicazioni. Il sistema operativo realizza due astrazioni concernenti i dischi: una è il dispositivo a basso livello, un semplice array di blocchi di dati; l'altra è il file system. Se il file system è relativo a un disco magnetico il sistema operativo accoda e organizza le richieste provenienti da diverse applicazioni. Nel seguito si espone il comportamento del sistema operativo nel trattare i supporti di memorizzazione rimovibili.

12.9.2.1 Interfaccia per le applicazioni

La maggior parte dei sistemi operativi gestisce i dischi rimovibili quasi come i dischi fissi. Quando s'inserisce un nuovo disco nella relativa unità, esso deve essere formattato, quindi si crea sul disco rimovibile un file system vuoto che si usa proprio come il file system di un'ordinaria unità a disco.

La gestione dei nastri, invece, è spesso differente; il sistema operativo di solito presenta un nastro come un supporto di memorizzazione a basso livello. Un'applicazione non apre un file presente nel nastro: apre l'intera unità nastro come dispositivo a basso livello. In questo caso, di solito, l'unità a nastro si riserva per l'uso esclusivo da parte di tale applicazione fino a che essa termina o chiude il dispositivo. L'esclusività è ragionevole perché l'accesso diretto ai dati presenti in un nastro può richiedere decine di secondi o persino qualche minuto, sicché intercalare gli accessi diretti a un nastro determinerebbe probabilmente enormi tempi d'accesso per un ridottissimo lavoro utile.

Quando un'unità a nastro è presentata come dispositivo a basso livello, il sistema operativo non fornisce i servizi del file system: è l'applicazione che deve decidere come usare l'array di blocchi. Un programma che crea una copia di riserva di un disco su un nastro potrebbe ad esempio scrivere una lista dei nomi e delle dimensioni dei file all'inizio del nastro, e poi copiare i dati dei file sul nastro in quell'ordine.

È facile rendersi conto dei problemi che possono sorgere quando si usi l'unità a nastro in questo modo: visto che ogni applicazione stabilisce i propri criteri di organizzazione del nastro, un nastro contenente dati può essere generalmente usato solo dal programma che lo ha creato. Se anche si sapesse, ad esempio, che un nastro con copie di riserva di file contiene una lista dei nomi e delle dimensioni dei file seguita dai dati dei file in quell'ordine, ci sarebbero comunque difficoltà nell'uso del nastro: non è nota l'esatta maniera in cui i nomi dei file sono registrati, né se le dimensioni siano espresse nel codice binario o ASCII, o ancora se i file siano scritti uno per blocco o invece concatenati assieme in una lunghissima sequenza di byte. Non è nota neanche la dimensione dei blocchi del nastro, perché questo è un parametro che si può fissare indipendentemente per ogni blocco scritto.

Le operazioni fondamentali relative a un'unità a disco sono `read()`, `write()` e `seek()`; per le unità a nastro s'impiega un insieme di operazioni fondamentali diverso. Anziché usare l'operazione `seek()`, si usa l'operazione `locate()`. Si tratta di un'operazione più precisa dell'operazione `seek()` per il disco, perché posiziona il nastro in corrispondenza di uno specifico blocco logico, e non di un'intera traccia: localizzare il blocco 0 equivale a riavvolgere il nastro.

Nella maggior parte delle unità nastro è possibile localizzare qualunque blocco sia stato scritto in un nastro, ma se il nastro non è ancora completamente pieno non si può eseguire un'operazione `locate()` nell'area vuota oltre l'area registrata; ciò poiché la maggior parte delle unità a nastro gestisce lo spazio fisico diversamente dalle unità a disco. I settori di un disco hanno una dimensione fissa, e si deve usare il processo di formattazione per assegnare la posizione definitiva ai settori vuoti prima che possano contenere qualunque informazione. La maggior parte delle unità a nastro ha una dimensione dei blocchi variabile, e la dimensione di ogni blocco si determina al momento della scrittura del blocco in questione. Se durante la scrittura s'incontra una regione difettosa del nastro, la si salta e si riscrive il blocco. Tutto ciò spiega perché non sia possibile compiere un'operazione `locate()` nella regione di nastro vuota presente oltre l'area già registrata: le posizioni e la numerazione dei blocchi logici non sono ancora state determinate.

Nella maggior parte dei casi le unità a nastro dispongono di un'operazione `read_position()` che riporta il numero del blocco logico in corrispondenza del quale si trova la testina. Molte unità a nastro hanno anche un'operazione `space()` per gli spostamenti relativi: l'operazione `space(-2)`, ad esempio, riavvolge il nastro di due blocchi logici.

In molti tipi di unità a nastro la scrittura di un blocco produce come effetto collaterale la cancellazione logica di tutto ciò che si trova oltre la posizione della scrittura. Ciò significa che nella maggior parte dei casi le unità a nastro sono dispositivi a solo accodamento di dati (*append-only devices*); in altre parole, l'aggiornamento di un blocco posto in mezzo al nastro comporta la cancellazione di tutto ciò che segue tale blocco. L'unità a nastro realizza l'accodamento scrivendo un simbolo di fine nastro (*end of tape*, EOT) dopo l'ultimo blocco registrato: l'unità rifiuta di compiere un'operazione `locate()` oltre il simbolo EOT, ma può localizzare l'EOT stesso e poi cominciare a scrivere. Quest'azione ha l'effetto di sovrascrivere il simbolo EOT e di accodarne uno nuovo alla fine dei blocchi appena scritti.

In linea di principio si potrebbero realizzare file system per i nastri ma, poiché le unità a nastro sono dispositivi a solo accodamento, molte strutture dati e algoritmi sarebbero diversi da quelli che si usano per i dischi.

12.9.2.2 Nomi dei file

Un'altra questione che il sistema operativo deve affrontare è l'assegnazione dei nomi ai file residenti nei mezzi rimovibili. Nel caso di un disco fisso ciò non è difficile: nei PC i nomi dei file consistono di una lettera rappresentante un'unità seguita da un nome di percorso; nel sistema operativo UNIX i nomi dei file non contengono riferimenti alle unità, ma la tabella di montaggio permette al sistema operativo di identificare l'unità contenente ciascun file. Se però il disco è rimovibile, il fatto che un'unità abbia ospitato un certo disco non facilita la ricerca del file. Se ogni disco rimovibile avesse un numero di serie diverso, un file residente in un'unità potrebbe presentare come prefisso il proprio numero di serie, ma in questo caso sarebbe necessario usare circa 12 cifre per evitare che due dischi possano avere lo stesso numero di serie: non è pensabile che gli utenti ricordino numeri di 12 cifre per identificare i loro file.

Le cose si complicano ulteriormente nel caso s'intenda scrivere dati su un supporto rimovibile in un certo calcolatore e poi riutilizzare lo stesso mezzo in un altro calcolatore: se entrambi i calcolatori sono dello stesso tipo e hanno lo stesso tipo di unità, l'unico problema è quello di conoscere i contenuti e l'organizzazione dei dati contenuti nel mezzo in questione; ma se i calcolatori o le unità sono di diverso tipo possono sorgere molte altre difficoltà. Anche se le unità fossero compatibili, calcolatori diversi potrebbero memorizzare i dati secondo ordini diversi, o usare codifiche diverse per i numeri binari e persino per le lettere (ASCII nei PC, EBCDIC nei mainframe).

In genere gli attuali sistemi operativi lasciano irrisolto il problema dei nomi per i mezzi rimovibili, confidando nel fatto che le applicazioni o gli utenti forniranno una chiave di lettura e di interpretazione dei dati. Per fortuna alcuni tipi di mezzi rimovibili sono così ben standardizzati da essere usati allo stesso modo da tutti i calcolatori. Un esempio è dato dai CD: i CD musicali sono registrati in un formato universalmente noto e leggibile da ogni riproduttore di CD. I CD di dati sono disponibili in pochi formati diversi, ed è normale che sia l'unità di lettura sia il sistema operativo siano in grado di gestirli. Anche i formati dei DVD sono ben standardizzati.

12.9.2.3 Gestione gerarchica della memoria

Un **juke-box** automatizzato permette a un calcolatore di cambiare un nastro o un disco rimovibile senza l'intervento di un utente. Due tra le principali applicazioni di questa tecnica sono relative alla realizzazione di copie di riserva e ai sistemi di gestione gerarchica della memoria. L'uso dei juke-box per la creazione di copie di riserva è molto semplice: quando un nastro o un disco sono pieni, il calcolatore richiede al juke-box di passare al successivo. Alcuni juke-box contengono decine di unità e migliaia di nastri, con bracci automatizzati che spostano i nastri nelle unità.

Un sistema di gestione gerarchica della memoria estende la gerarchia di memorizzazione oltre la memoria centrale e secondaria (cioè, i dischi magnetici) comprendendo la memoria terziaria; quest'ultima è di solito costituita di un juke-box di nastri o di dischi rimovibili. Si tratta del livello di memoria meno costoso e più capiente, ma probabilmente anche più lento.

Sebbene il sistema della memoria virtuale si possa estendere senza difficoltà alla memoria terziaria, in pratica ciò avviene raramente; infatti recuperare dati per mezzo di un juke-box può richiedere decine di secondi o addirittura minuti: attese così lunghe sono inconciliabili con le tecniche di paginazione su richiesta e con altri modi d'uso della memoria virtuale.

La tecnica più comune per estendere la gerarchia di memorizzazione fino alla memoria terziaria consiste nell'ampliare il file system. I file piccoli e frequentemente usati rimangono nei dischi magnetici, mentre i file vecchi, ingombranti e raramente necessari si archiviano nel juke-box. In alcuni sistemi per l'archiviazione dei file gli elementi di directory corrispondenti ai nomi dei file continuano a comparire nelle directory anche dopo l'archiviazione, ma i contenuti dei file non occupano più spazio nella memoria secondaria. Quando un'applicazione tenta di aprire un file archiviato, la chiamata di sistema `open()` rimane sospesa finché i contenuti del file possono essere reinstallati dalla memoria terziaria; una volta nuovamente disponibili nei dischi magnetici, la `open()` restituisce il controllo all'applicazione, e quest'ultima può ora accedere ai contenuti del file.

Al giorno d'oggi, la **gestione gerarchica della memoria secondaria** (*hierarchical storage management*, HSM) si applica solitamente a sistemi con grandi quantità di dati usati di rado, sporadicamente o periodicamente. Le ricerche correnti in questo campo tentano di estendere HSM con una piena **gestione del ciclo di vita delle informazioni** (*information life-cycle management*, ILM). I dati si spostano dai dischi ai nastri e di nuovo ai dischi, a seconda delle necessità, e sono cancellati secondo una scaletta o in accordo con una certa strategia. Alcuni siti, ad esempio, conservano i messaggi di posta elettronica per sette anni, ma pretendono l'assicurazione che dopo questo periodo siano distrutti. Allo scadere dei sette anni, i dati rilevanti potrebbero risiedere sui dischi, sui nastri HSM o sui nastri per le copie di riserva.Una buona ILM centralizza le informazioni sulla collocazione dei dati. Ciò permette di applicare la stessa strategia di gestione a specifici insiemi di dati, indipendentemente dallo loro collocazione fisica.

12.9.3 Prestazioni

Così come avviene per ogni componente del sistema operativo, i tre aspetti più importanti riguardanti le prestazioni della memorizzazione terziaria sono velocità, affidabilità e costi.

12.9.3.1 Velocità

La velocità della memoria terziaria è definita da due fattori: ampiezza di banda e latenza. La prima si misura in byte al secondo; in particolare, l'**ampiezza di banda sostenuta** è la velocità media di trasferimento nel caso di una rilevante quantità di dati – in altre parole, il numero di byte diviso il tempo di trasferimento; l'**ampiezza di banda effettiva** è invece il numero di byte trasferiti rapportato al tempo di I/O totale, inclusi il tempo richiesto da una `seek()` o una `locate()` e l'attesa eventualmente dovuta a cambi di dischi o nastri eseguiti dal juke-box. Essenzialmente, l'ampiezza di banda sostenuta è la velocità di trasferimento nel momento in cui i dati stanno effettivamente fluendo, mentre l'ampiezza di banda effettiva è la velocità di trasferimento complessiva fornita dall'unità. Con l'espressione *ampiezza di banda di un'unità* s'intende generalmente l'ampiezza di banda sostenuta.

L'ampiezza di banda per le unità a dischi rimovibili varia da pochi megabyte al secondo nei tipi più lenti, a più di 40 MB al secondo nei più veloci. Le unità a nastro hanno un'ampiezza di banda simile, da pochi megabyte fino a 30 MB al secondo.

Il secondo fattore è la **latenza d'accesso**. Rispetto a questo parametro i dischi sono molto più veloci dei nastri: la memorizzazione nei dischi è essenzialmente bidimensionale – tutti i bit sono, per così dire, all'aperto; un accesso al disco si compie semplicemente spostando il braccio verso il cilindro selezionato e aspettando che il settore interessato ruoti sotto la testina, il che può avvenire in meno di 5 millisecondi. Per contro, la memorizzazione nei nastri è tridimensionale: a ogni dato istante solo una piccola parte del nastro è accessibile alla testina, mentre il resto dei bit è sepolto sotto centinaia o migliaia di strati di nastro avvolto in una bobina. Un accesso diretto a un nastro richiede lo svolgimento o il riavvolgimento della bobina finché il blocco richiesto raggiunge la testina, cosa che può richiedere decine o centinaia di secondi. Si può quindi affermare in linea generale che l'accesso diretto a un nastro è oltre mille volte più lento dell'accesso diretto a un disco.

Se in tutto ciò è coinvolto anche un juke-box la latenza d'accesso può crescere notevolmente: per cambiare un disco rimovibile l'unità deve fermare la rotazione del motore, il braccio automatizzato deve scambiare i dischi, e l'unità deve avviare la rotazione del nuovo disco. Questa operazione richiede parecchi secondi, un tempo pari a circa cento volte il tempo medio d'accesso diretto a un disco singolo: lo scambio di dischi in un juke-box comporta una penalizzazione relativamente alta delle prestazioni.

Il tempo impiegato da un braccio automatizzato nel caso dei nastri è circa lo stesso che nel caso dei dischi; in genere, però, prima di poter essere estratto dall'unità un nastro deve essere completamente riavvolto, e quest'operazione può richiedere anche 4 minuti. Inoltre, dopo che un nuovo nastro è stato caricato, l'unità può aver bisogno di molti secondi per calibrarsi rispetto al nuovo nastro e prepararsi all'I/O. Sebbene un lento juke-box per nastri possa richiedere 1 o 2 minuti per scambiare i nastri, questo tempo non è sproporzionatamente lungo rispetto al tempo necessario per l'accesso diretto a un singolo nastro.

Generalizzando, quindi, si può dire che l'accesso diretto a un disco in un juke-box ha una latenza dell'ordine delle decine di secondi, mentre nel caso dei nastri in un juke-box la latenza è dell'ordine delle centinaia di secondi; lo scambio dei dischi è oneroso, mentre non lo è quello dei nastri. Si tratta ovviamente di un discorso di carattere generale: alcuni costosi juke-box per nastri sono capaci di riavvolgere ed estrarre un nastro, caricarne uno nuovo e posizionarlo a uno specifico punto impiegando complessivamente meno di 30 secondi.

Se si considerano soltanto le prestazioni delle unità di lettura e scrittura di un jukebox, i tempi di latenza e l'ampiezza di banda appaiono ragionevoli; non appena si concentra l'attenzione sui mezzi rimovibili si riscontra però un tremendo collo di bottiglia per le prestazioni. Si consideri in primo luogo l'ampiezza di banda: il rapporto fra l'ampiezza della banda e la capacità di memorizzazione di una biblioteca automatizzata è molto più sfavorevole di quello di un disco fisso. La lettura di tutti i dati contenuti in un disco di grandi dimensioni potrebbe richiedere circa un'ora, ma leggere tutti i dati memorizzati in un ingombrante archivio di nastri potrebbe richiedere anni. Per ciò che riguarda la latenza d'accesso la situazione è quasi altrettanto grama: se 100 richieste sono in coda per un'unità a disco fisso, il tempo d'attesa medio sarà di circa 1 secondo, ma se 100 richieste sono in coda per un archivio di nastri, il tempo d'attesa medio potrebbe essere più di 1 ora. La convenienza economica della memoria terziaria è dovuta alla possibilità di usare molte cartucce (a disco o a nastro) a basso costo con poche costose unità di lettura e scrittura. Un archivio di supporti rimovibili, però, è soprattutto adatto alla registrazione di dati usati raramente, perché il numero di richieste di I/O soddisfacibili per ogni ora d'uso di un tale archivio è relativamente basso.

12.9.3.2 Affidabilità

Nonostante si tenda a identificare il concetto di *buone prestazioni* con quello di *alta velocità*, un altro importante aspetto delle prestazioni è l'*affidabilità*: se non si riuscisse a leggere dati a causa di un guasto dell'unità o del supporto di memorizzazione, il tempo d'accesso sarebbe infinitamente lungo e l'ampiezza di banda infinitamente bassa. È quindi importante analizzare l'affidabilità dei mezzi rimovibili.

I dischi magnetici rimovibili sono meno affidabili dei dischi fissi, perché è più probabile che siano esposti a condizioni ambientali dannose come polvere, sbalzi di temperatura o umidità e forze meccaniche come urti o piegature. I dischi ottici sono considerati molto affidabili, perché lo strato che memorizza le informazioni è protetto da uno strato trasparente di plastica o vetro. L'affidabilità dei nastri magnetici è molto variabile poiché dipende dal tipo di unità di lettura e scrittura: alcune unità poco costose consumano un nastro dopo averlo usato poche decine di volte, mentre altre sono così delicate da permettere il reimpiego del nastro milioni di volte. Rispetto alla testina di un'unità a disco, la testina di un'unità a nastro è meno affidabile: la prima è sospesa sopra il disco, mentre la seconda è a stretto contatto col nastro, e l'attrito conseguente può rovinarla dopo un elevato numero di ore d'uso.

Riassumendo si può dire che le unità a disco fisso sono più affidabili delle unità a nastri o a dischi rimovibili, e che i dischi ottici sono probabilmente più affidabili dei dischi e dei nastri magnetici. Anche le unità a disco fisso hanno punti deboli: la collisione della testina col disco in genere distrugge i dati, mentre il guasto di un'unità a nastro o di un'unità a dischi ottici lascia spesso intatto il supporto di memorizzazione in uso al momento del guasto.

12.9.3.3 Costi

Il costo della memoria è un altro fattore importante: l'esempio seguente mostra concretamente come i mezzi rimovibili possano ridurre i costi totali di memorizzazione. Si supponga che un'unità a disco di x GB costi 200 dollari; di questa cifra, 190 dollari sono dovuti al controllore, al motore e al contenitore esterno, e 10 dollari ai piatti magnetici. Il costo della memoria fornita da questa unità è quindi di $200/x$ dollari al gigabyte. Si supponga ora di poter incapsulare i piatti magnetici in un disco rimovibile: il costo complessivo di un'unità e di 10 dischi è allora di 190 + 100 dollari, e la capacità di memoria totale è di $10x$ GB, cosicché il costo per gigabyte scende a $29/x$ dollari al gigabyte. Sebbene il costo di produzione

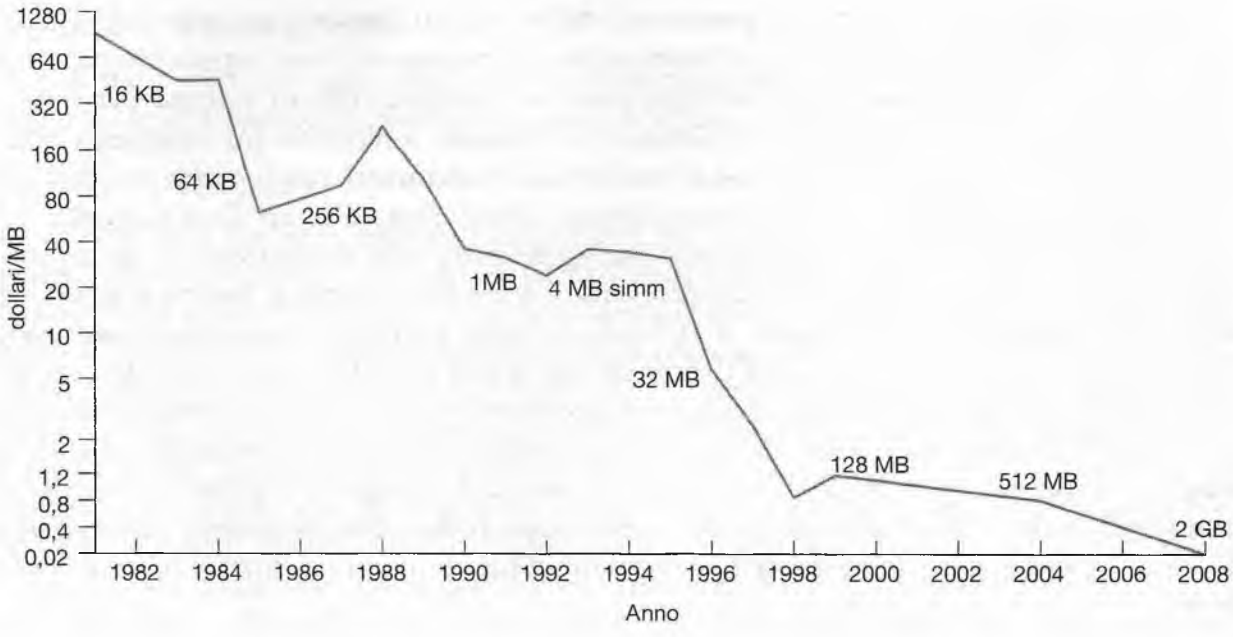


Figura 12.15 Prezzo al MB della memoria DRAM, dal 1981 al 2008.

delle unità a dischi rimovibili sia un po' più alto, la maggior spesa per un'unità è bilanciata dal basso costo di molte cartucce rimovibili, quindi il costo per gigabyte della memoria rimovibile può essere ben inferiore a quello delle unità a disco fisso.

Nelle Figure 12.15, 12.16 e 12.17 sono mostrate rispettivamente le tendenze dei prezzi al megabyte della memoria DRAM, delle unità a disco e delle unità a nastro. I prezzi riportati nei grafici sono i più bassi riscontrati fra tutte le inserzioni pubblicitarie apparse in varie riviste di settore e sul Web alla fine di ogni anno. Essi riflettono le caratteristiche del mercato dei piccoli calcolatori, oggetto dell'interesse dei lettori, i cui prezzi sono più bassi rispetto a minicalcolatori e mainframe. Nel caso dei nastri, il prezzo si riferisce a un'unità dotata di un solo nastro; poiché il costo di un nastro è in genere una piccola frazione del costo di un'unità, il

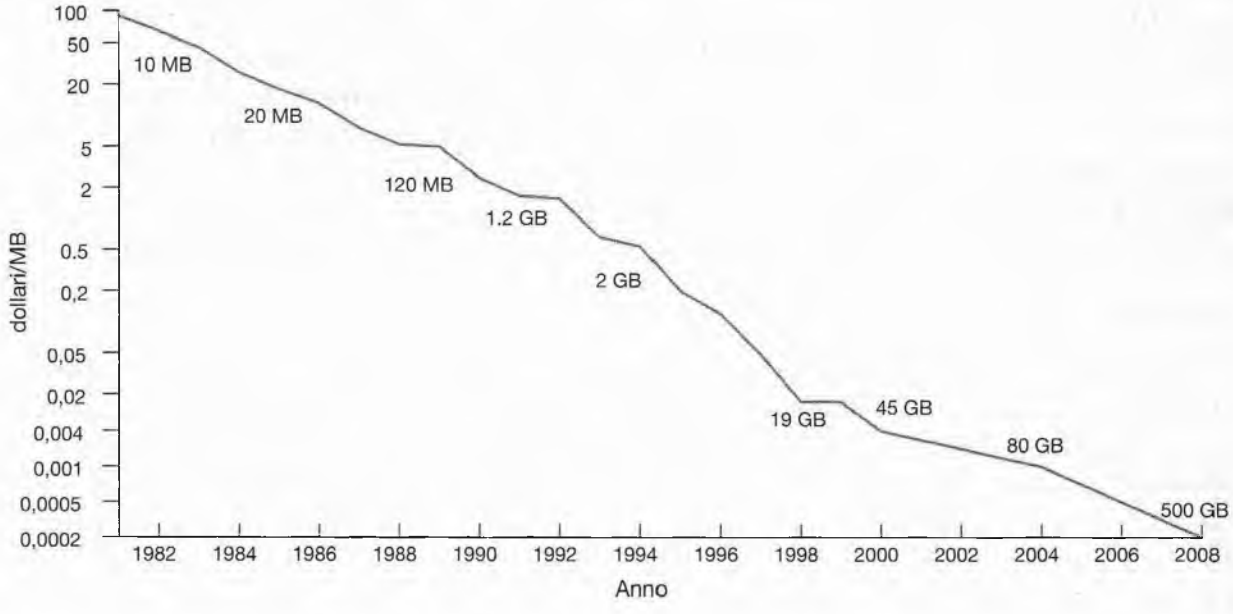


Figura 12.16 Prezzo al MB delle unità a disco magnetico, dal 1981 al 2008.

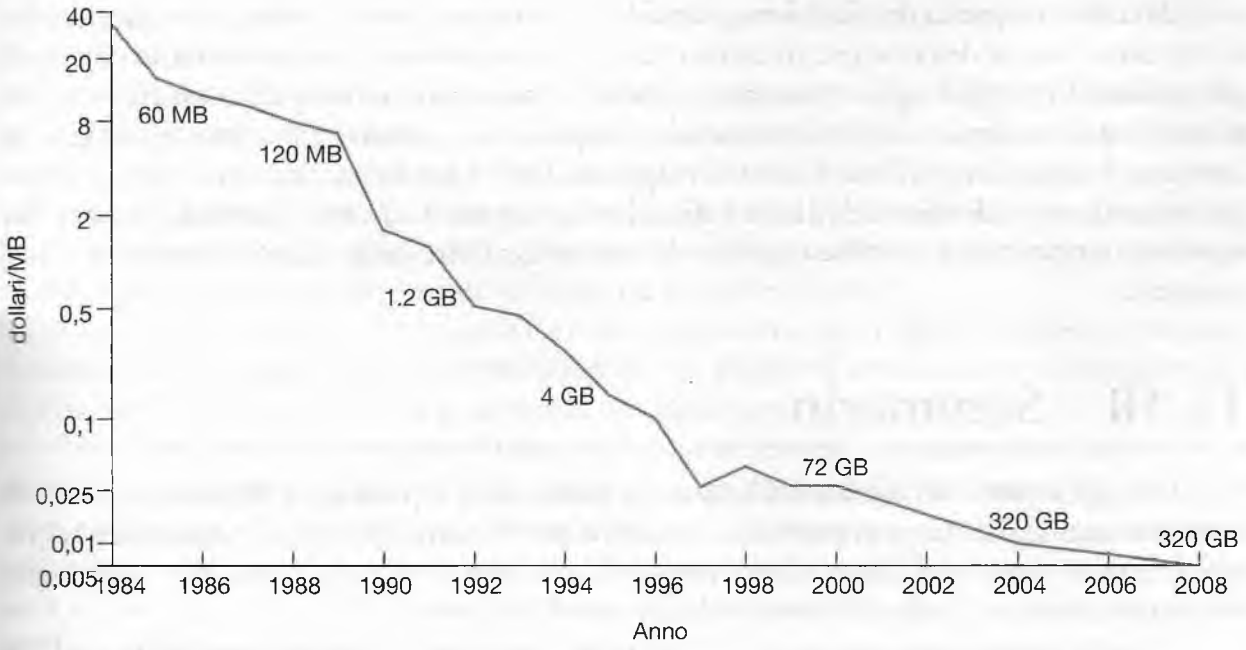


Figura 12.17 Prezzo al MB delle unità a nastro, dal 1984 al 2008.

costo totale della memorizzazione nei nastri diviene progressivamente molto più basso in seguito all'acquisto di nuovi nastri da usare in una stessa unità. Infatti in un grande archivio automatizzato contenente migliaia di nastri, il costo di memorizzazione è dominato dal costo dei nastri. Nel corso del 2004 il costo al GB dei nastri si aggirava intorno ai 40 dollari.

Come mostra la Figura 12.15, il costo della DRAM ha subito notevoli fluttuazioni: fra il 1981 e il 2004 si possono notare tre crolli del prezzo (attorno al 1981, al 1989 e al 1996) dovuti a una sovrapproduzione che causò la saturazione del mercato; si notano anche due periodi (attorno al 1987 e al 1993) durante i quali la scarsità d'offerta ha provocato un notevole aumento dei prezzi. Per quel che riguarda le unità a disco, (Figura 12.16), invece, la diminuzione dei prezzi è stata molto più regolare, anche se dal 1992 la corsa al ribasso ha avuto un'accelerazione. I prezzi delle unità a nastro sono scesi con regolarità fino al 1997 (Figura 12.17). In seguito il prezzo al GB delle unità a nastro economiche ha cessato di precipitare, sebbene i prezzi dei prodotti di tecnologie di medio livello, come i DAT/DDS, siano continuati a scendere, e attualmente siano vicini alle unità economiche. I prezzi delle unità a nastro sono indicati a partire dal 1984, perché, come già detto, la stampa di settore si occupa del mercato dei piccoli calcolatori, dei quali le unità a nastro non erano un accessorio comune prima di questa data.

Da questi grafici si nota quanto il prezzo della memorizzazione su disco sia crollato negli ultimi vent'anni; confrontando i grafici si deduce anche quanto, rispetto a quello delle memorie DRAM e dei nastri, si sia notevolmente abbassato.

Il prezzo al megabyte dei dischi magnetici si è ridotto di oltre quattro ordini di grandezza dal 1981 al 2004, mentre il prezzo corrispondente per gli elementi di memoria centrale è sceso di tre ordini di grandezza. La memoria centrale è attualmente 100 volte più costosa dei dischi.

Il prezzo al megabyte delle unità a dischi magnetici si avvicina al prezzo dei nastri magnetici, escludendo il costo delle unità a nastro. Ne consegue che gli archivi di nastri medi e piccoli hanno un costo di memorizzazione dei dati maggiore di un sistema a dischi di corrispondente capacità.

Il crollo dei prezzi dei dischi magnetici ha reso in gran parte obsoleta la memorizzazione terziaria: non si dispone più di alcuna tecnologia di memorizzazione terziaria che sia di più ordini di grandezza più economica dei dischi magnetici. Sembra che una ripresa della memorizzazione terziaria debba attendere un importante passo avanti in una tecnologia innovativa. Nel frattempo, l'uso dei nastri magnetici resterà per lo più limitato a scopi come la creazione di copie di riserva di dischi e di archivi composti di enormi quantità di nastri, che superano largamente l'effettiva capacità di memorizzazione delle grandi batterie di dischi magnetici.

12.10 Sommario

Per la maggior parte dei calcolatori le unità a dischi sono il principale dispositivo di I/O di memoria secondaria. La gran parte dei dispositivi per la memorizzazione secondaria usa i dischi o i nastri magnetici. Le moderne unità a disco sono strutturate come un grande array monodimensionale di blocchi logici, solitamente di 512 byte.

I dischi possono essere collegati al computer in due modi: (1) tramite le porte per l'I/O locali della macchina o (2) tramite connessioni di rete.

Le richieste di I/O sui dischi sono generate sia dal file system sia dai sistemi di memoria virtuale, e ognuna di esse specifica l'indirizzo cui fare riferimento nel disco sotto forma di numero di un blocco logico. Gli algoritmi di scheduling per i dischi possono aumentare l'ampiezza di banda, e ridurre il tempo di risposta medio e la variabilità del tempo di risposta. Algoritmi come l'SSTF, SCAN, C-SCAN, LOOK e C-LOOK sono progettati per realizzare questi miglioramenti tramite criteri di ordinamento della coda di richieste di operazioni sui dischi.

Le prestazioni possono essere influenzate negativamente dalla frammentazione esterna. Alcuni sistemi includono programmi rivolti a questo problema che esaminano l'intero file system alla ricerca di file frammentati, e spostano i blocchi tentando di ridurne la frammentazione. La deframmentazione di un file system può portare a un notevole incremento delle prestazioni, almeno quando esso si trova in avanzato stato di frammentazione; tuttavia, le prestazioni del sistema durante tale operazione si riducono. File system raffinati, come il Fast File System di UNIX, adottano molte tecniche per controllare la frammentazione durante l'allocazione dello spazio, rendendo superflua la riorganizzazione del disco.

Il sistema operativo gestisce i blocchi di un disco: innanzitutto si deve formattare fisicamente il disco per creare i settori direttamente sui suoi piatti – i dischi nuovi sono generalmente venduti già formattati; in seguito, il disco può essere diviso in partizioni, si può creare il file system, e si possono assegnare i blocchi d'avviamento che conterranno il programma d'avviamento del sistema; infine, quando un blocco diviene difettoso, il sistema deve essere in grado di isolarlo o di sostituirlo logicamente con un blocco di riserva.

Poiché un'area d'avvicendamento efficiente è essenziale per un sistema dalle buone prestazioni, molti sistemi usano un accesso di basso livello per l'I/O di paginazione. Certi sistemi riservano all'area d'avvicendamento una partizione di basso livello, altri impiegano un ordinario file all'interno del file system. Altri sistemi lasciano la scelta fra queste due possibilità all'utente o all'amministratore di sistema.

A causa della quantità di spazio per la registrazione dei dati richiesta nei grandi sistemi, i dischi sono spesso resi ridondanti tramite algoritmi RAID. Questi algoritmi permettono l'uso di più dischi per una data operazione, e consentono la prosecuzione del funzionamento del sistema e anche il ripristino automatico dei dati a fronte del guasto di un disco.

Gli algoritmi RAID sono classificati in livelli che offrono diverse combinazioni di affidabilità ed elevate velocità di trasferimento.

Lo schema di registrazione con scrittura anticipata (*write-ahead logging*) richiede la disponibilità di spazio di memorizzazione stabile, che va implementato replicando le informazioni necessarie su memorie non volatili multiple (di solito dischi) i cui guasti siano indipendenti gli uni dagli altri. È anche necessario aggiornare le informazioni in modo controllato, sì da assicurare la possibilità di recuperare i dati stabili a seguito di qualsivoglia malfunzionamento durante il trasferimento o il recupero dei dati.

La memoria terziaria è realizzata da unità a nastro o a disco capaci di operare con mezzi rimovibili. Le tecnologie disponibili sono molte; alcune di esse sono i nastri magnetici, i dischi rimovibili magnetici e magneto-ottici e i dischi ottici.

Nel caso dei dischi rimovibili il sistema operativo fornisce in genere tutti i servizi di un file system, compresa la gestione dello spazio e lo scheduling delle richieste di I/O. Per molti sistemi operativi il nome di un file residente in una cartuccia rimovibile è una combinazione del nome dell'unità e del nome di un file contenuto in quell'unità; questa convenzione è più semplice, ma anche potenzialmente più ambigua, dell'uso di un numero di serie per l'identificazione di una specifica cartuccia.

Nel caso dei nastri il sistema operativo in genere fornisce semplicemente un'interfaccia a basso livello. Molti sistemi operativi, inoltre, non incorporano metodi specifici per la gestione dei juke-box: questi ultimi devono essere controllati da un driver di dispositivo apposito o da un'applicazione che si occupa della creazione delle copie di riserva o della HSM.

Tre importanti fattori delle prestazioni sono l'ampiezza di banda, la latenza e l'affidabilità. Esistono unità a dischi e nastri con un'ampia gamma di diverse ampiezze di banda, ma la latenza d'accesso dei nastri è generalmente molto più elevata di quella dei dischi. Anche lo scambio di cartucce eseguito da un juke-box è relativamente lento; visto poi che per un juke-box il rapporto fra unità e cartucce è basso, la lettura di una gran parte dei dati archiviati può richiedere un tempo molto lungo. I dischi ottici sono in generale più affidabili di quelli magnetici, perché nei primi lo strato fisico contenente le informazioni è protetto da due strati di materiale trasparente, mentre nei secondi il materiale magnetico è maggiormente esposto a eventuali danni. Infine, negli ultimi vent'anni il costo della memoria è diminuito sensibilmente, in particolare per quanto riguarda la memorizzazione su disco.

Esercizi pratici

12.1 Lo scheduling del disco, fatta eccezione per l'FCFS, è utile in un ambiente con un solo utente? Argomentate.

12.2 Spiegate perché l'SSTF tende a favorire i cilindri centrali rispetto a quelli più interni e più esterni.

12.3 Perché la latenza di rotazione non viene solitamente presa in considerazione nello scheduling del disco? Come potreste modificare lo scheduling SSTF, lo SCAN e quello C-SCAN per includervi l'ottimizzazione della latenza?

12.4 In che modo l'utilizzo di un disco RAM potrebbe condizionare la vostra scelta di un algoritmo di scheduling del disco? Quali fattori dovrebbero essere considerati? Gli stessi fattori sono applicabili anche allo scheduling dell'unità a disco, dato che il file system memorizza blocchi usati recentemente in una buffer cache nella memoria centrale?

12.5 Perché è importante bilanciare gli I/O del file system tra i dischi e i controllori su un sistema in un ambiente multitasking?

12.6 Quali tradeoff implica la rilettura delle pagine di codice da un file system rispetto all'utilizzo dell'area di avvicendamento per memorizzarli?

12.7 Esiste un modo per realizzare una memorizzazione delle informazioni veramente stabile? Argomentate.

12.8 Il termine "Fast Wide SCSI-II" denota un bus SCSI che opera a una velocità di dati di 20 megabyte al secondo mentre sposta pacchetti di byte tra un host e un dispositivo. Supponete che una unità a disco Fast Wide SCSI-II giri a 7200 RPM, abbia una dimensione dei settori di 512 byte e contenga 160 settori per traccia.

a. Stimate la velocità di trasferimento sostenuta da questa unità a disco in megabyte al secondo.

b. Supponete che l'unità abbia 7000 cilindri, 20 tracce per cilindro, un tempo di spostamento della testina (da un piatto all'altro) di 0,5 millisecondi e un tempo di ricerca del cilindro adiacente di 2 millisecondi. Utilizzate questa informazione per dare una stima accurata della velocità di trasferimento sostenuta per un trasferimento di grandi dimensioni.

c. Supponete che il tempo di ricerca medio per l'unità sia di 8 millisecondi. Stimate le operazioni di I/O al secondo e la velocità di trasferimento effettiva per un carico di lavoro ad accesso diretto che legge settori individuali distribuiti sul disco.

d. Calcolate il numero di operazioni di I/O ad accesso diretto al secondo e la velocità di trasferimento per I/O con dimensioni di 4, 8 e 64 kilobyte.

e. Se vi sono richieste multiple in coda, un algoritmo di scheduling come quello per scansione dovrebbe essere in grado di ridurre la distanza media di ricerca. Supponete che un carico di lavoro ad accesso diretto stia leggendo pagine di 8 kilobyte, la lunghezza media della coda sia 10, e l'algoritmo di scheduling riduca il tempo di ricerca medio a 3 millisecondi. Calcolate ora il numero di operazioni di I/O al secondo e la velocità di trasferimento effettiva dell'unità.

12.9 A un bus SCSI può essere connessa più di una unità a disco. In particolare, un bus Fast Wide SCSI-II (si veda Esercizio 12.8) può essere connesso al massimo a 15 unità a disco. Ricordate che questo bus ha un'ampiezza di banda di 20 megabyte al secondo. In ogni momento, solo un pacchetto può essere trasferito sul bus tra la cache interna del disco e l'host. Tuttavia, un disco può muovere il proprio braccio mentre altri dischi stanno trasferendo un pacchetto sul bus. Inoltre, mentre un disco sta trasferendo dati tra i propri piatti magnetici e la propria cache interna, altri dischi trasferiscono un pacchetto sul bus. In riferimento alle velocità di trasferimento che avete calcolato per i vari carichi di lavoro nell'Esercizio 12.8, discutete quanti dischi possono essere effettivamente utilizzati da un bus Fast Wide SCSI-II.

12.10 Rimappare blocchi difettosi tramite l'accantonamento o la traslazione di settori può influenzare le prestazioni. Supponete che l'unità dell'Esercizio 12.8 abbia in totale 100 settori difettosi in locazioni casuali e che ogni settore difettoso sia mappato in un settore di riserva posizionato su una traccia differente all'interno dello stesso cilindro. Stimate il numero di operazioni di I/O al secondo e la velocità di trasferimento effettiva per un carico di lavoro ad accesso diretto che consiste in letture da 8 kilobyte, assumendo che la lunghezza di coda sia 1 (e quindi che la scelta dell'algoritmo di scheduling sia ininfluente). Qual è l'effetto di un settore difettoso sulle prestazioni?

12.11 Quale effetto si produce in un juke-box di dischi se il numero dei file aperti è superiore rispetto al numero delle unità contenute nel juke-box?

12.12 Se un disco fisso magnetico avesse lo stesso costo al gigabyte di un nastro, i nastri diventerebbero obsoleti o sarebbero ancora necessari? Spiegate la vostra risposta.

12.13 A volte un nastro è detto mezzo ad accesso sequenziale, mentre un disco magnetico è considerato un mezzo ad accesso diretto. In realtà, l'idoneità di un dispositivo di memorizzazione all'accesso diretto dipende dalla grandezza del trasferimento. Il termine *velocità di trasferimento in streaming* denota la velocità di un trasferimento di dati che è in corso, escluso l'effetto della latenza di accesso. La *velocità effettiva di trasferimento*, invece, è il rapporto dei byte totali per il totale dei secondi, incluso il tempo di overhead come la latenza di accesso. Ipotizzate che, in un computer, la cache di livello 2 abbia una latenza di accesso di 8 nanosecondi e una velocità di trasferimento in streaming di 800 megabyte al secondo, che la memoria principale abbia una latenza di accesso di 60 nanosecondi e una velocità di trasferimento in streaming di 80 megabyte al secondo, che il disco magnetico abbia una latenza di accesso di 15 millisecondi e una velocità di trasferimento in streaming di 5 megabyte al secondo, e che una unità a nastro abbia una latenza di accesso di 60 secondi e una velocità di trasferimento in streaming di 2 megabyte al secondo.

a. L'accesso diretto causa la diminuzione della velocità effettiva di trasferimento di un dispositivo, perché non vengono trasferiti dati durante il tempo di accesso. Per il disco descritto, qual è la velocità di trasferimento effettiva se l'accesso medio è seguito da un trasferimento in streaming di (1) 512 byte, (2) 8 kilobyte, (3) 1 megabyte, e (4) 16 megabyte?

b. L'utilizzo di un dispositivo è definito come il rapporto tra la velocità effettiva di trasferimento e la velocità di trasferimento in streaming. Calcolate l'utilizzo dell'unità a disco per ognuna delle quattro grandezze di trasferimento indicate al punto a.

c. Supponete che un utilizzo del 25 per cento o più sia considerato accettabile. Utilizzando i valori di prestazione indicati, calcolate la dimensione minima di trasferimento per un disco che dia un utilizzo accettabile.

d. Completate la frase seguente: Un disco è un dispositivo ad accesso diretto per trasferimenti superiori a _____ byte ed è un dispositivo ad accesso sequenziale per trasferimenti inferiori.

e. Calcolate le dimensioni minime di trasferimento che diano utilizzi accettabili per cache, memoria e nastro.

f. Quando un nastro può essere considerato un dispositivo ad accesso diretto e quando invece è un dispositivo ad accesso sequenziale?

12.14 Supponete di concordare sul fatto che 1 kilobyte sia 1024 byte, 1 megabyte sia 1024^2 byte e che 1 gigabyte sia 1024^3 byte. La serie continua con terabyte, petabyte ed esabyte (1024^6). Diversi progetti scientifici prevedono di arrivare a registrare e a memorizzare qualche esabyte di dati nei prossimi dieci anni. Per rispondere alle seguenti domande, dovrete fare un certo numero di ipotesi ragionevoli: enunciatele esplicitamente.

a. Quante unità a disco sarebbero necessarie per contenere 4 esabyte di dati?

b. Quanti dischi magnetici sarebbero necessari per contenere 4 esabyte di dati?

c. Quanti nastri ottici sarebbero necessari per contenere 4 esabyte di dati (si veda l'Esercizio 12.35)?

d. Quante cartucce olografiche di memorizzazione sarebbero necessarie per contenere 4 esabyte di dati (si veda l'Esercizio 12.34)?
e. Quanti centimetri cubi di spazio di memorizzazione richiederebbe ciascuna delle opzioni citate?

Esercizi

12.15 Eccetto l'FCFS, nessuno tra i criteri di scheduling del disco descritti è veramente *equo* (si può avere un blocco indefinito).

a. Spiegate perché questa affermazione è vera.
b. Descrivete una maniera di modificare gli algoritmi come lo SCAN in modo che risultino equi.
c. Spiegate perché l'equità è un obiettivo importante in un sistema a partizione del tempo.
d. Date tre o più esempi di circostanze in cui è importante che il sistema operativo sia iniquo nel servire le richieste di I/O.

12.16 Supponete che un'unità a disco abbia 5000 cilindri numerati da 0 a 4999. L'unità serve attualmente una richiesta relativa al cilindro 143, e la richiesta precedente era relativa al cilindro 125. La coda di richieste inevase, in ordine FIFO, è composta di richieste riguardanti i cilindri

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Assumendo come punto di partenza la posizione attuale della testina, calcolate la distanza totale (in cilindri) che il braccio del disco percorre per soddisfare tutte le richieste inevase usando i seguenti algoritmi di scheduling:

a. FCFS;
b. SSTF;
c. SCAN;
d. LOOK;
e. C-SCAN;
f. C-LOOK.

12.17 È noto dalla fisica elementare che quando un oggetto è sottoposto a un'accelerazione costante a, la relazione fra la distanza d e il tempo t è data da $d = 1/2\ at^2$. Supponete che l'unità a disco dell'Esercizio 12.16, durante la ricerca di un cilindro, imprima al braccio del disco un'accelerazione costante per la prima metà del tragitto richiesto dalla ricerca, e imprima invece una decelerazione costante e della stessa intensità dell'accelerazione precedente per la seconda metà del tragitto. Ipotizzate che l'unità possa portare a termine in 1 millisecondo la ricerca di un cilindro adiacente, e in 18 millisecondi una ricerca a tutto raggio lungo i 5000 cilindri.

a. La distanza di una ricerca è il numero di cilindri attraverso i quali la testina deve passare. Spiegate perché il tempo di ricerca è proporzionale alla radice quadrata della distanza percorsa.
b. Scrivete il tempo di ricerca in funzione della distanza da percorrere. L'equazione dovrebbe avere la forma $t = x + y\sqrt{L}$, dove t è il tempo in millisecondi ed l è la distanza da percorrere in cilindri.

c. Calcolate il tempo totale di ricerca per ogni algoritmo di scheduling dell'Esercizio 12.16 relativamente alla coda di richieste lì descritta. Determinate quale algoritmo sia più veloce, cioè implichi un tempo di ricerca totale minore.

d. L'*aumento percentuale di velocità* è il tempo risparmiato diviso il tempo originariamente necessario. Calcolate l'aumento percentuale di velocità dell'algoritmo più veloce rispetto all'FCFS.

12.18 Supponete che il disco dell'Esercizio 12.17 ruoti alla velocità di 7200 giri al minuto.

a. Calcolate la latenza di rotazione media di quest'unità a disco.

b. Dite quale distanza di ricerca è possibile coprire nel tempo calcolato al punto a).

12.19 Una ricerca di dati in accelerazione come quella descritta nell'Esercizio 12.17 è tipica delle unità a disco rigido. I floppy disk, invece, come pure molti dischi rigidi prodotti prima della metà degli anni '80, eseguono solitamente delle ricerche a velocità fissa. Supponete che il disco dell'Esercizio 12.17 abbia una ricerca a velocità costante piuttosto che una ricerca ad accelerazione costante, cosicché il tempo di ricerca sia formulabile come $t = x + yL$, dove t è il tempo in millisecondi mentre L è la distanza di ricerca. Supponete che il tempo per ricercare un cilindro adiacente sia di 1 millisecondo, come prima, mentre il tempo per ricercare ogni cilindro aggiuntivo sia di 0,5 millisecondi.

a. Scrivete un'equazione che esprima il tempo di ricerca in funzione della distanza di ricerca.

b. Utilizzando il punto a), calcolate il tempo di ricerca totale per ognuna delle richieste dell'Esercizio 12.16. La risposta ottenuta è uguale a quella dell'Esercizio 12.17(c)?

c. Qual è l'aumento percentuale di velocità dell'algoritmo di scheduling rispetto all'FCFS in questo caso?

12.20 Scrivete un programma che simuli gli algoritmi di scheduling del disco trattati nel Paragrafo 12.4.

12.21 Confrontate le prestazioni di SCAN e C-SCAN assumendo una distribuzione uniforme delle richieste di I/O. Considerate il tempo di risposta medio (cioè il tempo che intercorre fra l'arrivo di una richiesta e il completamento dell'operazione a essa associata), la variazione del tempo di risposta, e l'effettiva ampiezza di banda. Analizzate come le prestazioni dipendano dalle dimensioni relative del tempo di ricerca e della latenza di rotazione.

12.22 Le richieste non sono di solito uniformemente distribuite: ad esempio un cilindro che contiene FAT o inode del file system potrebbe essere visitato più spesso di un cilindro che contiene solo file. Supponete di sapere che il 50 per cento delle richieste sia relativo a un piccolo numero fisso di cilindri.

a. Dite se uno fra gli algoritmi illustrati in questo capitolo sia particolarmente adatto a questa circostanza. Motivate la risposta.

b. Proponete un algoritmo di scheduling che offra prestazioni anche migliori sfruttando questo "punto caldo" del disco.

c. Di solito i file system localizzano i blocchi di dati usando una tabella indiretta, come la FAT nell'MS-DOS o gli inode in UNIX. Descrivete uno o più modi di sfruttare queste tabelle per migliorare le prestazioni del disco.

12.23 Spiegate se e come sia possibile conseguire, per le richieste di lettura, migliori prestazioni con il sistema RAID a livello 1, piuttosto che con il sistema RAID a livello 0 (con il sezionamento non ridondante dei dati).

12.24 Considerate un sistema RAID a livello 5, comprensivo di cinque dischi; nel quinto disco risiedono le parità per gli insiemi di quattro blocchi di quattro dischi. A quanti blocchi si deve accedere per effettuare le operazioni seguenti:

a. scrittura di un blocco di dati;
b. scrittura di sette blocchi contigui di dati.

12.25 Ponete a confronto il throughput ottenuto attraverso un sistema RAID a livello 5 con quello ottenuto mediante un sistema RAID a livello 1, in merito a:

a. operazioni di lettura su blocchi singoli;
b. operazioni di lettura su blocchi multipli contigui.

12.26 Paragonate le prestazioni raggiunte da un sistema RAID a livello 5 con quelle di un sistema RAID a livello 1, relativamente alle operazioni di scrittura.

12.27 Assumete di avere una configurazione mista, che comprenda dischi strutturati secondo il sistema RAID a livello 1, e altri dischi a livello RAID 5. Supponente che il sistema, per memorizzare un determinato file, sia libero di optare per l'una o l'altra soluzione. Quali file dovrebbero essere memorizzati nei dischi a livello RAID 1 e quali nei dischi a livello RAID 5, allo scopo di ottimizzare le prestazioni?

12.28 L'affidabilità di un'unità a disco generalmente si quantifica usando il **tempo medio fra due guasti** (*mean time between failures*, MTBF); sebbene sia chiamata "tempo", questa quantità in effetti si misura in ore di funzionamento d'unità per guasto.

a. Dato un gruppo di 1000 unità a disco, ognuna delle quali ha un MTBF di 750.000 ore, dite con quale frequenza avverrà il guasto di un'unità del gruppo, scegliendo fra le seguenti possibilità quella che si adatta meglio alla situazione descritta: una volta ogni mille anni, una volta ogni cento anni, una volta ogni dieci anni, una volta al mese, una volta alla settimana, una volta al giorno, una volta ogni ora, una volta al minuto, una volta al secondo.
b. Le statistiche di mortalità indicano che in media un individuo residente negli Stati Uniti d'America ha circa 1 probabilità su 1000 di morire fra i 20 e i 21 anni d'età. Deducete l'MTBF di un ventenne, e convertite il risultato da ore in anni. Spiegate che cosa dice questo MTBF sulle aspettative di vita di un ventenne.
c. Un produttore asserisce che un certo modello di unità a disco abbia un MTBF di 1 milione di ore. Dite cosa si può concludere circa il numero di anni per cui una di queste unità a disco è coperta dalla garanzia.

12.29 Esponete vantaggi e svantaggi relativi all'accantonamento dei settori e alla traslazione dei settori.

12.30 Esponete i motivi per cui il sistema operativo potrebbe necessitare di informazioni accurate sulle modalità di memorizzazione dei blocchi sul disco. In termini di miglioramento delle prestazioni, in che modo il sistema operativo può mettere a frutto queste informazioni?

12.31 In genere il sistema operativo tratta i dischi rimovibili come un file system condiviso, ma assegna l'unità a nastro a una sola applicazione alla volta. Elencate tre moti-

vi che spieghino questa disparità di trattamento. Descrivete le caratteristiche aggiuntive che il sistema operativo dovrebbe avere per poter gestire un juke-box di nastri come un file system condiviso. Dite se anche le applicazioni che condividono il juke-box devono avere particolari caratteristiche, o se possono usare i file come se risiedessero in un disco; motivate la risposta.

12.32 Dite che effetto avrebbe sul costo e sulle prestazioni una densità superficiale dei dati nei nastri pari a quella dei dischi. (Con "densità superficiale" si intende il numero di gigabit per pollice quadrato.)

12.33 In questo esercizio userete alcune semplici stime per confrontare il costo e le prestazioni dei due seguenti sistemi: un sistema di memorizzazione con una capacità dell'ordine del terabyte interamente costituito di dischi; e un altro sistema che sfrutta la memoria terziaria. Supponete che ogni disco magnetico abbia una capacità di 10 GB, un costo di 1000 dollari, una velocità di trasferimento di 5 MB al secondo, e una latenza d'accesso media di 15 millisecondi. Supponete inoltre che un archivio di nastri costi 10 dollari per gigabyte, trasferisca 10 MB di dati al secondo e abbia una latenza d'accesso media di 20 secondi. Calcolate il costo totale, la massima velocità totale di trasferimento e il tempo d'attesa medio per il sistema a soli dischi. Se fate qualche ipotesi sul carico di lavoro, descrivetele e giustificatele. Supponete adesso che il 5 per cento dei dati sia usato di frequente e risieda quindi in dischi, ma che il restante 95 per cento sia memorizzato nell'archivio di nastri; supponete inoltre che il sistema dei dischi gestisca il 95 per cento delle richieste, e l'archivio di nastri il restante 5 per cento. Calcolate il costo totale, la massima velocità totale di trasferimento e il tempo d'attesa medio per questo sistema di gestione gerarchica della memoria.

12.34 Immaginate che qualcuno inventi un dispositivo di memorizzazione olografica. Supponete che il suo costo sia di 10.000 dollari, il tempo medio d'accesso di 40 millisecondi e che impieghi cartucce da 100 dollari della dimensione di un CD. Ogni cartuccia può contenere 40.000 immagini, e ciascuna immagine è un quadrato in bianco e nero composto di 6000×6000 pixel (ogni pixel memorizza il valore di un bit). Supponete infine che il dispositivo possa leggere o scrivere un'immagine in 1 millisecondo. Analizzate i seguenti aspetti:

a. i possibili usi di un tale dispositivo;
b. l'impatto di un tale dispositivo sulle prestazioni dell'I/O di un sistema;
c. l'eventuale possibilità che altri dispositivi di memorizzazione divengano obsoleti.

12.35 Supponete che un disco ottico di 5,25 pollici abbia una densità superficiale di dati di 1 gigabit per pollice quadrato. Supponete inoltre che un nastro magnetico abbia una densità superficiale di dati pari a 20 megabit per pollice quadrato, sia largo 1/2 pollice e lungo 1800 piedi. Stimate la capacità di memorizzazione dei due supporti. Ora immaginate che esista un nastro ottico delle stesse dimensioni fisiche del nastro magnetico descritto, ma con la stessa densità superficiale di dati del disco ottico. Calcolate la quantità di dati memorizzabile nel nastro ottico; posto che il nastro magnetico descritto costi 25 dollari, fissate un prezzo di mercato per il nastro ottico.

12.36 Valutate i modi in cui un sistema operativo potrebbe mantenere una lista dello spazio libero relativa a un file system residente in un nastro. Supponete che il dispositivo a nastro permetta il solo accodamento, e che usi il simbolo EOT e le operazioni `locate()`, `space()` e `read_position()` descritte nel Paragrafo 12.9.2.1.

12.11 Note bibliografiche

Le batterie ridondanti di dischi (RAID) sono presentate da [Patterson et al. 1988] e nella dettagliata rassegna di [Chen et al. 1994]. Le architetture delle unità a disco per elaborazioni ad alte prestazioni sono trattate da [Katz et al. 1989]. Approfondimenti sui sistemi RAID sono offerti da [Wilkes et al. 1996] e [Yu et al. 2000]. [Teorey e Pinkerton 1972] presentano una delle prime analisi comparate degli algoritmi di scheduling del disco; usano un modello di unità a disco con tempo di ricerca lineare rispetto al numero di cilindri attraversati. Per questo tipo di disco l'algoritmo LOOK è una buona scelta per code di richieste inevase lunghe meno di 140 elementi, mentre l'algoritmo C-LOOK lo è per code che superano i 100 elementi. [King 1990] illustra alcuni modi per ridurre il tempo di ricerca, ad esempio con lo spostamento del braccio quando l'unità a disco sarebbe altrimenti inattiva. [Seltzer et al. 1990], [Jacobson e Wilkes 1991] descrivono algoritmi di scheduling incentrati sulla latenza di rotazione, oltre che sui tempi di ricerca. L'ottimizzazione dello scheduling per impiegare al meglio i tempi morti del disco è trattata da [Lumb et al. 2000]. [Worthington et al. 1994] esamina le prestazioni dei dischi, mostrando il trascurabile effetto sulle prestazioni della gestione dei blocchi difettosi. Lo sfruttamento dell'esistenza di un "punto caldo" al fine di ridurre i tempi di ricerca è stato considerato da [Ruemmler e Wilkes 1991], nonché da [Akyurek e Salem 1993]. [Ruemmler e Wilkes 1994] descrivono un accurato modello delle prestazioni di una moderna unità a disco. [Worthington et al. 1995] spiega come determinare caratteristiche del disco di basso livello quali la struttura delle aree all'interno dei cilindri; questo lavoro è ulteriormente sviluppato in [Schindler e Gregory 1999]. Problemi connessi alla gestione della potenza del disco sono esaminati in [Douglis et al. 1994], [Douglis et al. 1995], [Greenawalt 1994], [Golding et al. 1995].

La dimensione dei trasferimenti richiesti e la casualità del carico di lavoro hanno un'influenza considerevole sulle prestazioni dell'unità a disco. [Ousterhout et al. 1985], e Ruemmler e [Wilkes 1993] riportano numerose interessanti caratteristiche dei carichi di lavoro, ad esempio che la maggior parte dei file sono brevi, la maggior parte dei file creati di recente è eliminata assai presto, che il più delle volte i file aperti per lettura sono letti in modo sequenziale nella loro interezza, e che nella maggior parte dei casi le distanze di ricerca sono brevi. [McKusick et al. 1984] descrive il Berkeley Fast File System (FFS), che adotta molte tecniche raffinate aventi lo scopo di ottenere buone prestazioni per parecchi tipi di carichi di lavoro. [McVoy e Kleiman 1991] illustra ulteriori miglioramenti dell'FFS originario. [Quinlan 1991] presenta la realizzazione di un file system per mezzi WORM con cache su dischi magnetici; [Richards 1990] propone un approccio alla memoria terziaria basato sul concetto di file system. [Maher et al. 1994] traccia una panoramica dell'integrazione di file system distribuiti e memoria terziaria.

Il concetto di gestione gerarchica della memorizzazione è studiato da più di un quarto di secolo: un articolo di [Mattson et al. 1970], ad esempio, descrive un metodo matematico per prevedere le prestazioni di un sistema di gestione gerarchica della memorizzazione. [Alt 1993] descrive l'integrazione della memoria rimovibile in un sistema operativo commerciale, mentre [Miller e Katz 1993] descrivono le caratteristiche dell'accesso alla memoria terziaria in un ambiente basato su di un supercalcolatore. [Benjamin 1990] fornisce una panoramica dei problemi connessi alla necessità di un vastissimo spazio di archiviazione per il progetto EOSDIS della NASA. La gestione e l'uso dei dischi collegati in rete, nonché dei dischi programmabili sono tematiche analizzate da [Gibson et al. 1997b], [Gibson et al. 1997a], [Riedel et al. 1998] e [Lee e Thekkath 1996].

La tecnologia della memorizzazione olografica è l'argomento dell'articolo di [Psaltis e Mok 1995]; una raccolta di articoli su quest'argomento, a partire dal 1963, è stata curata da [Sincerbox 1994]. [Asthana e Finkelstein 1995] descrivono diverse tecnologie di memorizzazione emergenti, compresa la memoria olografica, i nastri ottici e tecniche basate sul confinamento dell'elettrone. [Toigo 2000] offre un'approfondita descrizione delle moderne tecnologie dei dischi e di diverse future potenziali tecniche di registrazione dei dati.