Chapter 11

大量儲存器結構

章節目標

- 描述各種輔助儲存器的實體結構，以及裝置結構對其使用的影響
- 說明大量儲存器的效能特性
- 評估 I/O 排班演算法
- 討論作業系統提供大量儲存器的服務，包括 RAID

11.1 大量儲存器結構的概觀

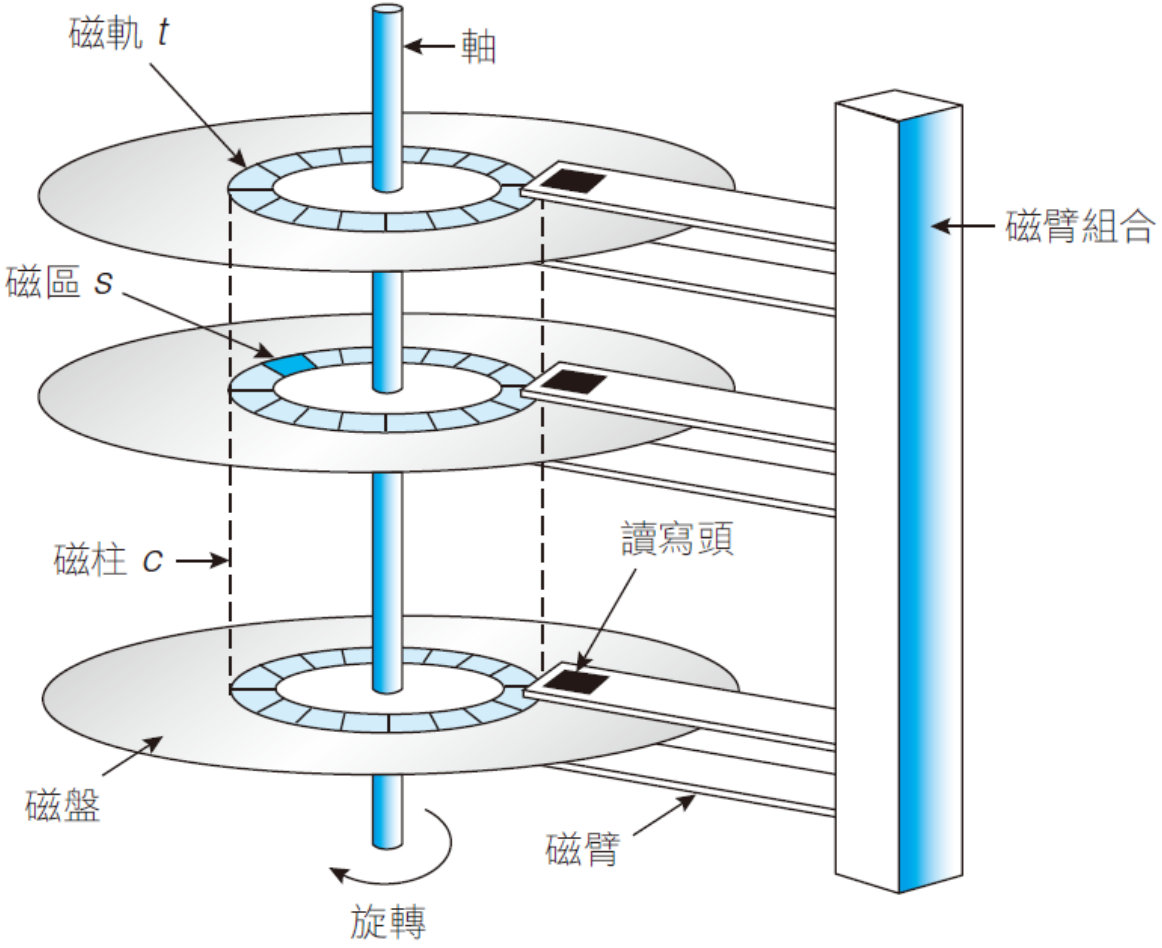


圖 11.1 移動磁頭的磁碟機制

11.1.1 硬碟機

- 從概念上來說，HDD 相對簡單 (圖 11.1)
  - 每個磁碟**磁盤** (platter) 具有扁平的圓形形狀類似 CD
  - 一般的磁盤直徑大小為 1.8 到 3.5 吋
  - 磁盤的兩個表面均覆蓋有磁性材料
  - 利用磁性的方式將資料記錄在磁盤上來儲存訊息，並透過檢測磁盤上的磁性型態來讀取訊息

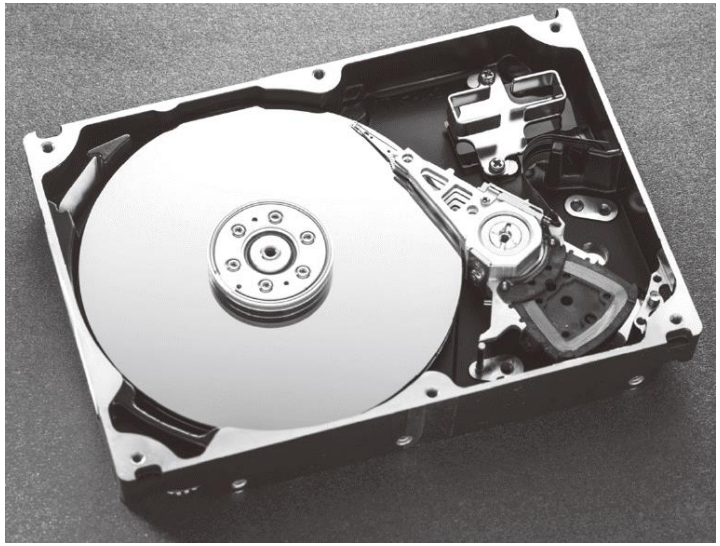


圖 11.2 拆除外殼的 3.5 吋硬碟

11.1.1 硬碟機

- 當磁碟機運轉時，磁碟機馬達以高速旋轉，大部份磁碟機的馬達轉速約為每秒 60 到250 轉間
  - 通常以**每分鐘多少轉** (rotation per minute, RPM) 為單位
  - 一般磁碟機的轉速是5,400、7,200、10,000 和 15,000 RPM
  - 某些磁碟機不使用時會關掉電源，當收到 I/O 請求時就會開啟並旋轉，傳輸速率跟旋轉速度有關，磁碟速度可以分為兩部份
  - **傳輸速率** (transfer rate) 為資料在硬碟與電腦間傳遞的速率

11.1.1 硬碟機

- **定位時間** (positioning time) 或稱為**隨機存取時間** (random-access time)
  - 包含移動磁臂到所在磁柱所需的時間稱為**搜尋時間** (seek time)，以及磁頭轉到所在磁區所需的時間稱為**旋轉潛伏期** (rotational latency)
- 通常磁碟每秒能傳輸幾百萬位元的資料，並且需要花費幾毫秒的搜尋時間和旋轉潛伏期
  - 它們通過在磁碟控制器中配備 DRAM 緩衝區來提高性能

11.1.1 硬碟機

- 由於磁頭是在一個非常薄的空氣或其它氣體，如氦的墊子上 (單位為微米) 快速飛行，於是可能會發生磁頭與磁碟表面接觸的風險
- 雖然磁盤已覆蓋一層薄的保護層，但有時磁頭還是會損壞磁性表面，這種意外稱為**磁頭損壞** (head crash)
- 磁頭損壞是不能修復，而是整個磁碟必須進行更換，而且除非硬碟上的資料有備份至其它的儲存空間或是被 RAID 功能保護，要不然硬碟裡的資料就會發生遺失

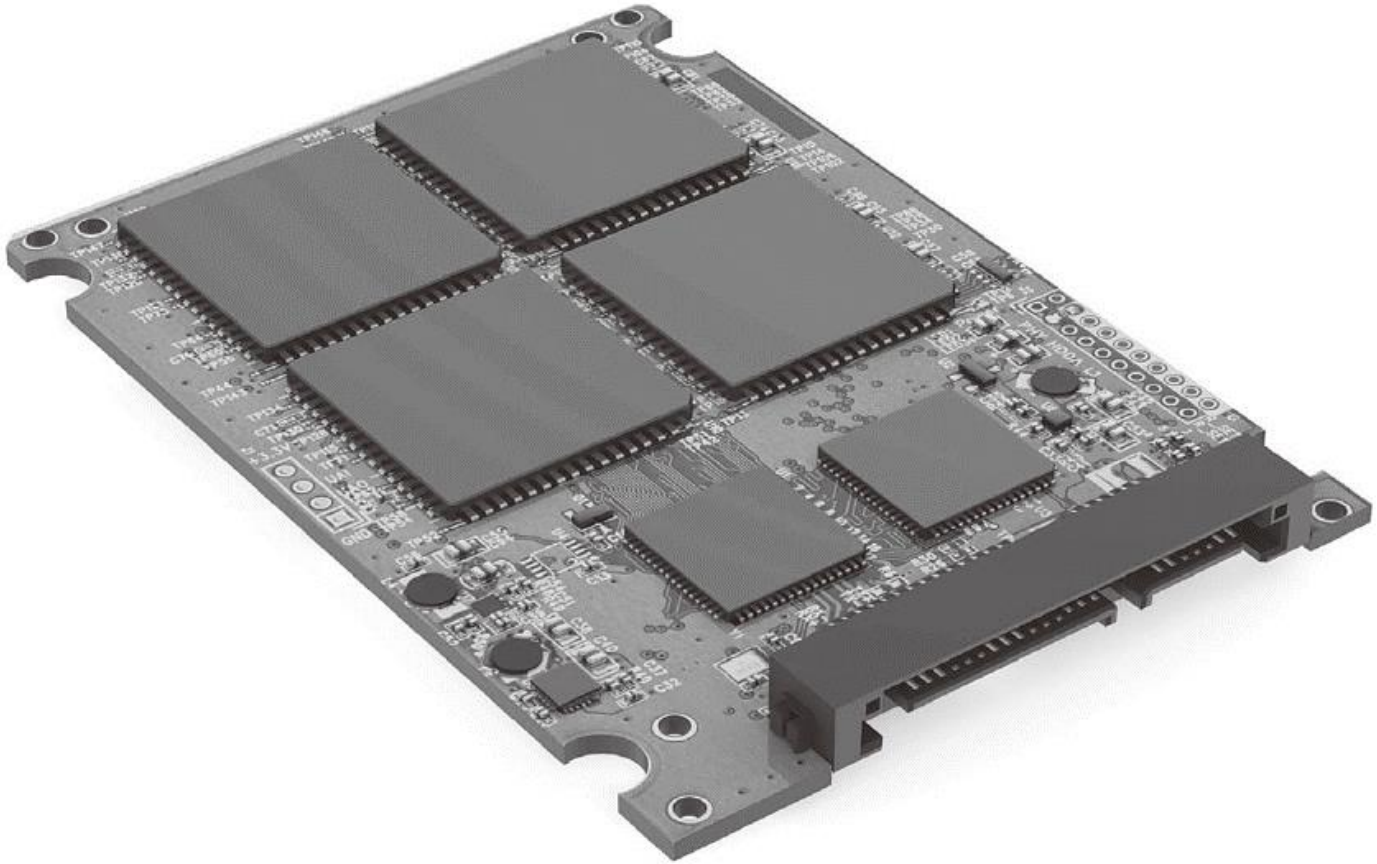
11.1.1 硬碟機

- 硬碟機是密封式的單元，而有些安裝可插拔的硬碟機外殼可以讓你不用移除機殼或是關閉系統，就能夠直接移除硬碟
- 這個做法相當有用，當系統需要的儲存量超過給定時間內連接所需的數量時，或是需要更換工作中的故障裝置
  - 其它類似的**可移除式** (removable) 儲存器媒介形式，包括 CD、DVD 和藍光磁碟機等

非揮發性記憶體裝置概述

- 基於快閃記憶體的非揮發性記憶體通常用於類似磁碟機機殼中，稱為**固態硬碟** (solidstate disk, SSD) (圖 11.3)
- 有時候它使用於 **USB 裝置** (USB drive) (也稱為拇指碟或隨身碟) 或 DRAM 條，它也能直接安裝在主機板上為主要儲存裝置
  - 如智慧型手機就這樣使用
- 在所有形式下，它的用途和作法都是類似這樣的方式，我們對非揮發性記憶體的討論集中在這項技術上

圖 11.3　3.5 吋的固態硬碟電路板



非揮發性記憶體裝置概述

- 非揮發性記憶體因為沒有移動性的元件，因此並沒有搜尋時間或旋轉潛伏期，與硬碟比較起來更加可靠
  - 這樣的方式消耗更少的功率
  - 非揮發性記憶體的缺點是每兆位元組比傳統硬碟貴，並且容量相較硬碟也較小
  - 但隨著時代演進，非揮發性記憶體的容量比起硬碟的容量成長更快，而價格下降也更快，因此它們的使用量正在很明顯地增加
  - 實際上固態硬碟和相關的裝置已經使用在某些筆記型電腦中，使其變得更小、更快與更節能

磁帶

- **磁帶** (magnetic tape) 被用作早期的輔助儲存媒介
  - 儘管它是非揮發性並且可以儲存大量資料，但是與主記憶器和驅動器相比，它的存取時間很慢
  - 隨機存取磁帶的速度比隨機存取硬碟機大約慢一千倍，比固態硬碟隨機存取慢約十萬倍
  - 因此磁帶對於輔助儲存不是很有用
  - 使用磁帶主要用於備份，儲存不常使用的資訊，以及作為將資訊從一個系統傳輸到另一個系統的媒介

磁帶

- 磁帶被保存在捲軸中，使用讀寫頭捲繞或重捲繞的方式轉動到磁帶上的正確位置可能需要幾分鐘
  - 但是一旦定位，磁帶裝置就可以與硬碟一樣的速度進行讀取和寫入資料
  - 量差異很大，主要取決於磁帶裝置的種類，目前的容量已超過幾 TB 空間
  - 某些磁帶機內建的壓縮功能可以讓儲存量增加一倍以上。磁帶及其驅動器通常依據寬度分類
    - 包括 4 毫米、8 毫米和 19 毫米和 1/4、l/2 吋
  - 有些是根據技術命名的，例如 LTO-6 (圖 11.5) 和 SDLT

圖 11.5　插入了磁帶匣的 LTO-6 磁帶裝置

11.1.4 輔助儲存器連接方法

- 輔助儲存裝置以一組稱為 **I/O 匯流排** (I/O bus) 的線連接到電腦上
  - 常用幾種匯流排，包含
    - **進階技術連接** (advanced technology atta chment, ATA)
    - **串列進階技術連接** (serial ATA, SATA)
    - **外部串列進階技術連接** (External serial ATA)
    - **串列式傳輸介面技術** (serial attached SCSI, SAS)
    - **萬用串列匯流排** (universal serial bus, USB)
    - **光纖通道** (fiberchannel, FC)
  - 最常使用的連接方法是 SATA

11.1.4 輔助儲存器連接方法

- 因為非揮發性記憶體裝置比硬碟來得快，工廠特地客製化一個快速介面，叫作非揮發性記憶體主機控制器介面規範 (NVM express, NVMe)
- NVMe 和其他連接方法相比，直接連接 PCI 匯流排跟裝置，可以增加傳輸量

- 資料經由**控制器** (controller) 或**主機匯流排轉接器** (host-bus adapter, HBA) 的特殊電子處理器在匯流排上進行傳遞
  - **主機控制器** (host controller) 是在匯流排的電腦終端控制器
  - **裝置控制器** (device controller) 是用來連結每一個儲存裝置

11.1.4 輔助儲存器連接方法

- 為執行大量儲存 I/O 運作，電腦執行一個命令到主機控制器，通常使用記憶體映射 I/O 埠
- 主機控制器經由訊息傳送命令到裝置控制器，而且裝置控制器操作磁碟機硬體來執行這個命令
- 通常裝置控制器有一個內建快取，磁碟機上的資料傳遞發生在快取和儲存媒介之間，並且在快取與主機控制器之間以快速電子速度將資料傳遞到主機通過直接記憶體存取 (DMA)在暫存主機 DRAM 之間出現

11.1.5 位址映射

- 儲存裝置被定址為一個大型一維陣列**邏輯區塊** (logical blocks)，其中邏輯區塊是最小的傳輸單位
  - 每個邏輯區塊都映射到實體磁區或半導體分頁
  - 一維邏輯區塊陣列映射到裝置的磁區或分頁上
  - 磁區 0 為硬碟上最外的第一條磁軌的第一磁區
    - 舉例來說，映射會按順序從最外面到最內側磁軌，經過該磁柱上的其餘磁軌，然後經過磁柱的其餘部分
  - 揮發性記憶體從晶片上的多元組、區塊、頁面映射到邏輯區塊的陣列裡

11.1.5 位址映射

- **邏輯區塊位址** (logical block address, LBA) 是比磁區、磁柱、標頭多元組 (head tuple) 或晶片、區塊、分頁多元組 (page tuple) 更簡單的演算法

11.2 硬碟排班

- 作業系統的工作就是有效率地使用硬體
  - 就 HDD 而言，滿足此責任意味著有快速的存取時間與最大資料傳輸頻寬
- 對於使用磁盤的 HDD 和其它機械儲存設備而言，存取時間有兩個主要的部份
  - 磁臂將讀寫頭移動到包含想要磁區之磁軌所需的時間，稱為搜尋時間 (seek time)
  - 旋轉潛伏期是因為等待將磁碟中想要的磁區旋轉到讀寫頭所在的位置而產生的
  - 裝置**頻寬** (bandwidth) 的定義為，傳送的位元組總數除以從第一個服務要求到最後傳送完成之間所需的總時間

11.2 硬碟排班

- 我們可以藉由採用一個好的順序，來對儲存 I/O 服務要求進行排班，以便改善存取時間和頻寬兩個因素
- 每當一個行程需要 I/O 對磁碟輸入或輸出資料時，就必須發出一個系統呼叫到作業系統
- 在這項要求中，應該指定一些必要的資訊：
  - 是輸入或輸出作業？
  - 開放檔案處理，指向要操作的檔案？
  - 要傳送的記憶體位址為何？
  - 要傳送多少資料量？

11.2 硬碟排班

- 如果想要使用的磁碟機或控制器可以使用，則請求可以被立刻服務
  - 如果裝置或控制器在忙錄中，任何新的系統請求會進行等待請求 (pending request) 佇列取代
  - 對於一個含有多個行程的多元程式系統而言，磁碟佇列通常可能有許多等待的佇列
- 已經存在的請求佇列藉由避免磁頭搜尋，讓裝置經由佇列管理增進效能的機會

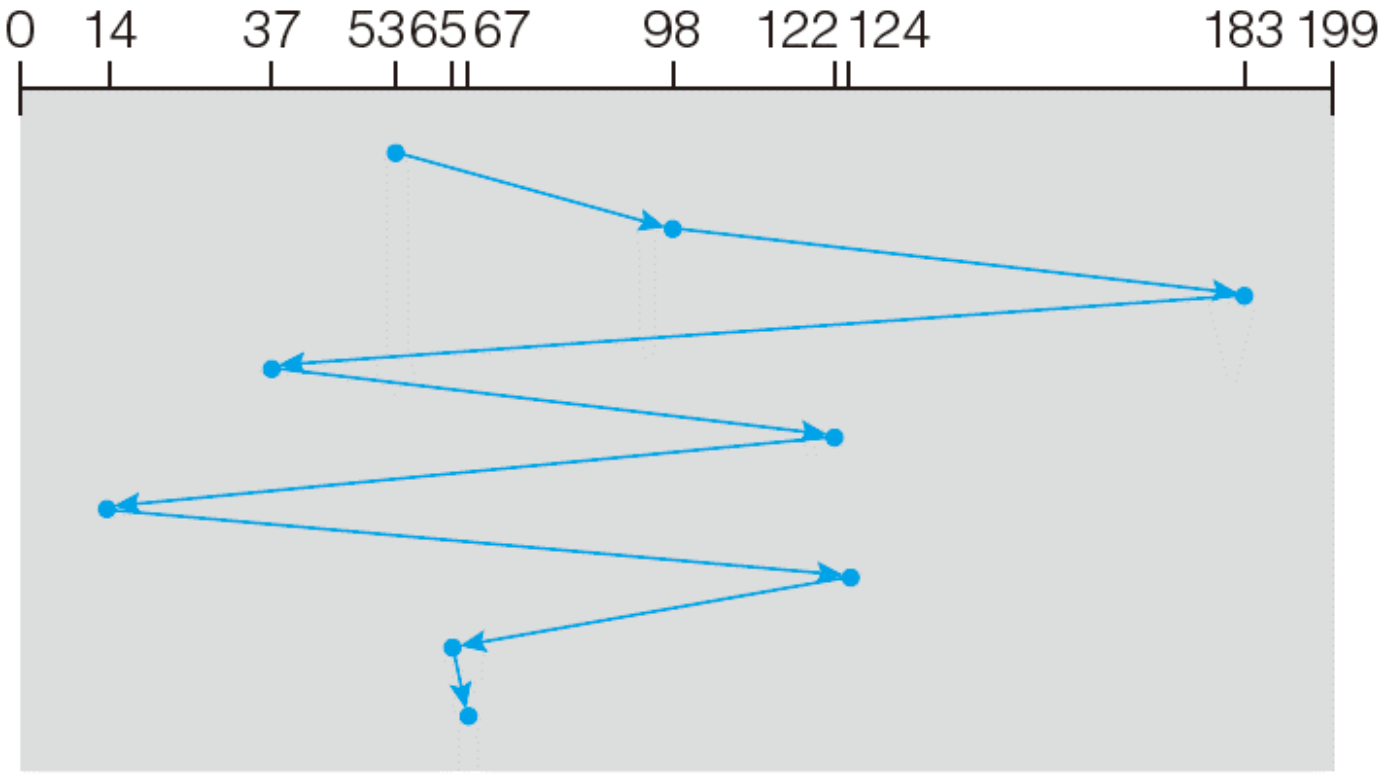
11.2.1 FCFS 排班

- 當然，先來先做 (first-come, first-served, FCFS) 是最簡單的一種排班演算法
  - 這種演算法本質上是公平的，但是一般而言，它不能提供最快的服務
  - 例如，考慮一個對於在磁柱上的區塊有許多 I/O 要求，如在磁碟佇列之中的以下排列：

    98、183、37、122、14、124、65、67
  - 如果磁碟讀寫頭的起始位址為磁柱 53
  - 因此磁頭共需移動 640 磁柱的距離
  - 此排序法如圖 11.6 所示

圖 11.6　FCFS 的磁碟排班

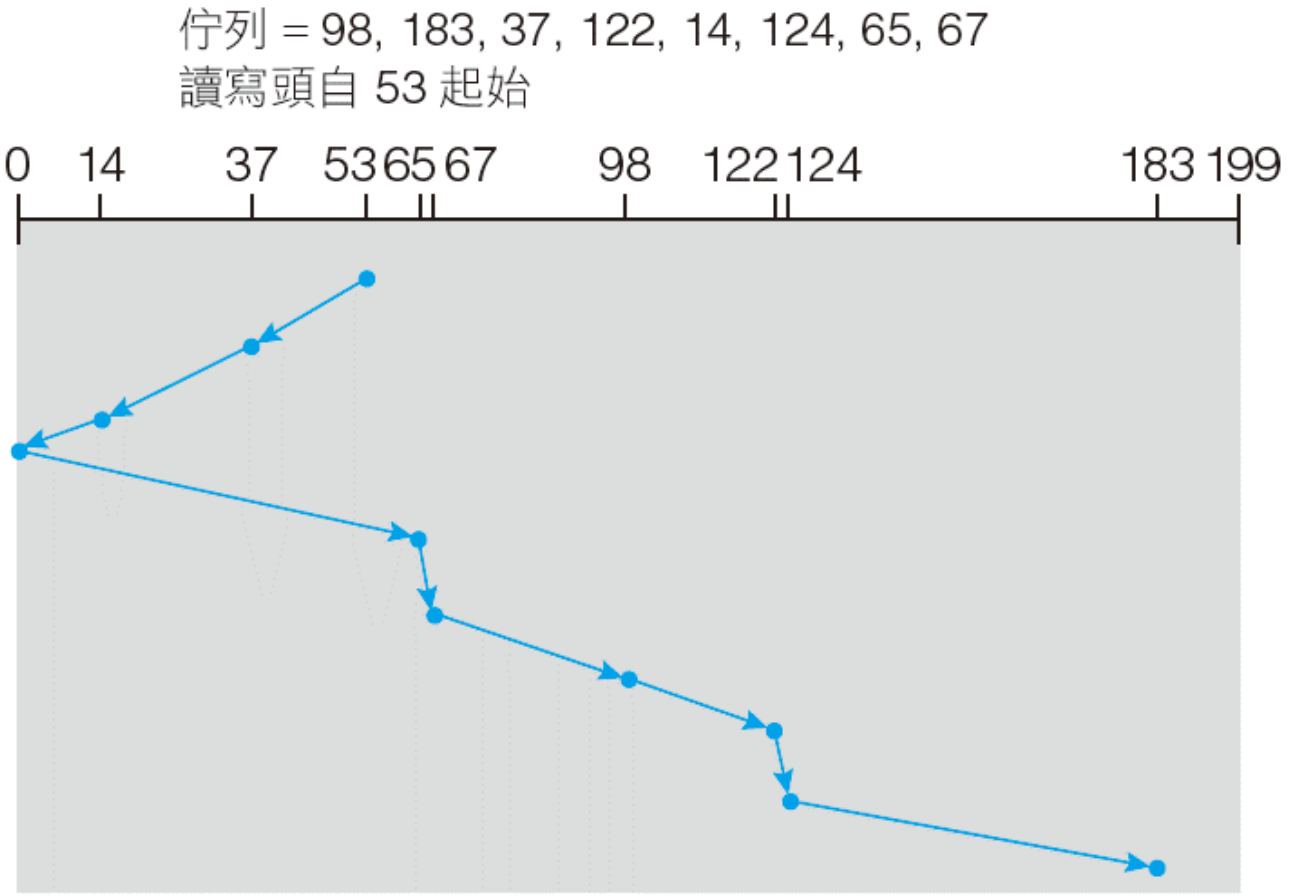
佇列 = 98, 183, 37, 122, 14, 124, 65, 67
讀寫頭自 53 起始



11.2.2 掃描排班

- 在 SCAN 演算法 (SCAN algorithm) 中，磁臂自磁碟的一端起始，並向另一端移動
  - 然後對其到達之每一要求服務的磁柱服務，直到磁碟的另一端為止
  - 再由另一端根據相反的順序移動磁頭，並繼續服務各要求，使得讀寫頭分別向前和向後跨越磁碟地連續掃描
  - 有時 SCAN 演算法又稱為**升降梯演算法** (elevator algorithm)
    - 因為磁臂的動作很像建築物中的一部電梯，首先服務所有向上的要求
    - 然後反向服務另一個相近的要求

圖 11.7　SCAN 磁碟排班

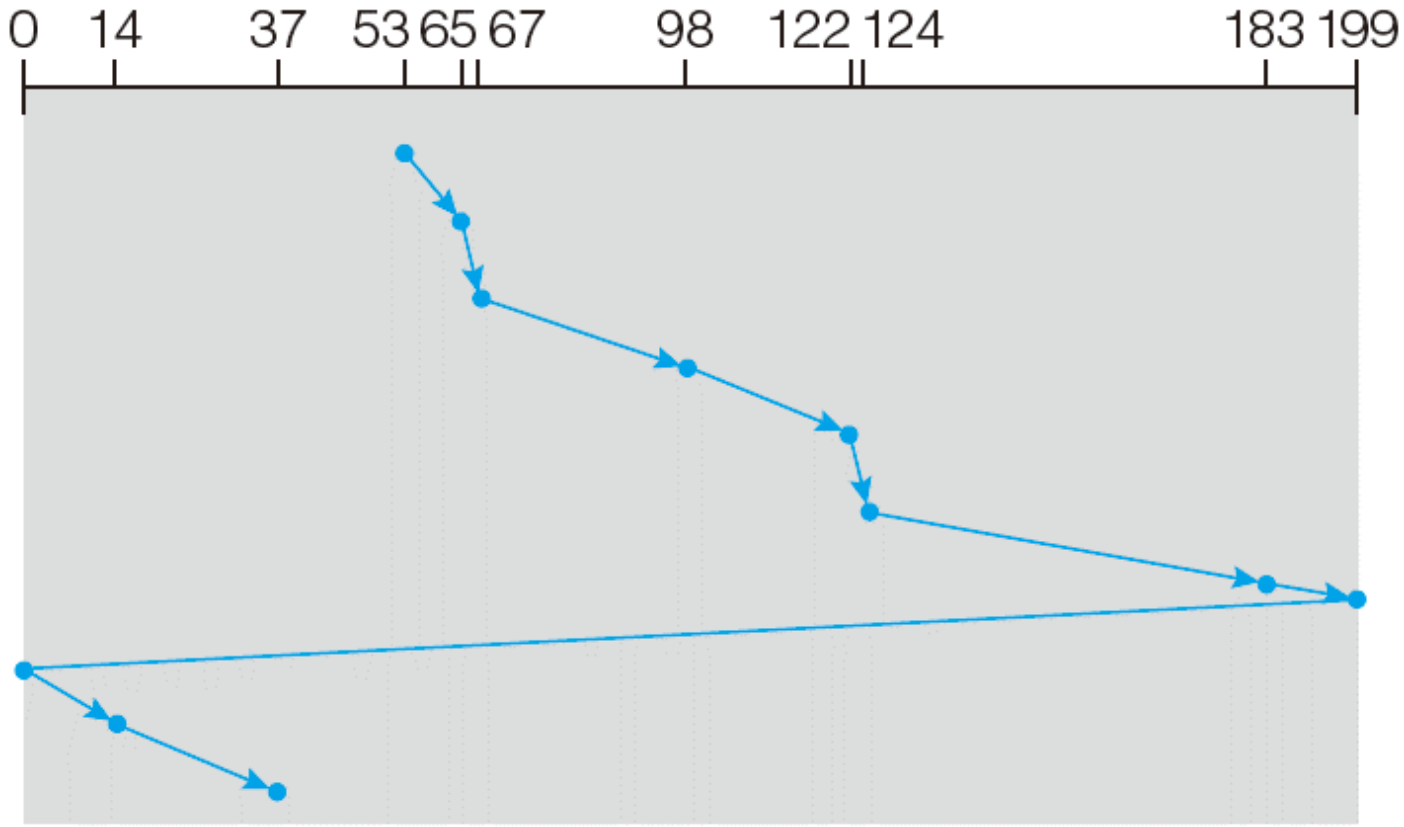


11.2.3 C-SCAN 排班

- **循環掃描排班** [circular SCAN (C-SCAN) scheduling] 是 SCAN 掃描法的變形，設計成提供更均勻的等候時間
  - 如同 SCAN，C-SCAN 也是將讀寫頭自磁碟的一端移到另一端，並執行經過的各個要求
  - 但當它到另一端時，就立即返回磁碟的起始點，而不在回程服務任何的要求

圖 11.8　C-SCAN 磁碟排班

佇列 = 98, 183, 37, 122, 14, 124, 65, 67
讀寫頭自 53 起始



11.2.4 磁碟排班演算法的選擇

- 有許多磁碟排班演算法並沒有被我們提到，因為它們很少被使用
- 但是作業系統設計師如何決定實施哪種方法，而部署者則選擇最佳使用方法？
  - 對於任何特定的請求列表，我們都可以定義最佳的檢索順序，但是最佳的排班演算法並不一定比 SCAN 公平
  - 對於任何演算法，效能都在很大程度上取決於請求的數量和類型
    - 例如，假設佇列通常只有一個未完成的請求，然後所有排班演算法都會相同，因為它們只有一個選擇位置來移動磁頭：

11.2.4 磁碟排班演算法的選擇

- 它們的行為都類似於 FCFS

- SCAN 和 C-SCAN 在磁碟上負載較重的系統上表現會更好，因為它們不太可能引起飢餓問題，但仍會存在該問題，這使得 Linux 建立截止期限 (deadline) 排班器
  - 該排班器維護單獨的讀取和寫入佇列，並賦予讀取優先權
  - 因為行程較可能對區塊寫入比讀取多
  - 佇列會依據 LBA 順序來排序，實際上實作了 C-SCAN
  - 所有 I/O 請求均依據此 LBA 順序批次發送。截止期限有四個佇列：

11.2.4 磁碟排班演算法的選擇

- 兩個讀取和兩個寫入
- 一個依據 LBA 排序
- 另一個依據 FCFS 排序

- 它在每個批次之後進行檢查，以查看 FCFS 佇列中是否存在比配置的使用期限 (預設為 500 毫秒) 更早的請求
- 如果是這樣，將會包含該請求的 LBA 佇列 (讀或寫) 選擇用於下一批次 I/O

11.5 儲存裝置管理

- 一個新的儲存裝置是全部空的狀態：
  - 它只是磁的記錄材料或一組未初始化的半導體儲存單元
  - 在儲存裝置可以儲存資料之前，它必須使得控制器可以區分磁區的讀和寫
- NVM 分頁必須初始化並建立 FTL，這個過程稱為**低階格式化** (low-level formatting)
  - 或**實體格式化** (physical formatting)
  - 低階格式化為每個儲存位置使用特殊的資料結構寫入裝置

11.5 儲存裝置管理

- 磁區/分頁的資料結構通常由標頭、資料區域和標尾組成
- 標頭和標尾包含控制器使用的訊息
  - 例如作為磁區/分頁，以及錯誤檢測或糾正代碼

11.5 儲存裝置管理

- 第一步是將磁碟**分割** (partition) 成一個或多個磁區或分頁，作業系統可以視每一個分割為一個分離的磁碟
  - 例如，一個分割可以保存作業系統的可執行碼的一個備份、一個保存置換空間，另一個則保存使用者檔案
    - 當檔案系統要管理整個裝置時，部份作業系統和檔案系統會自動執行分割
    - 分割資訊以固定的格式寫入儲存裝置上的固定位置
    - 在 Linux 中，fdisk 命令用於管理儲存裝置上的分割

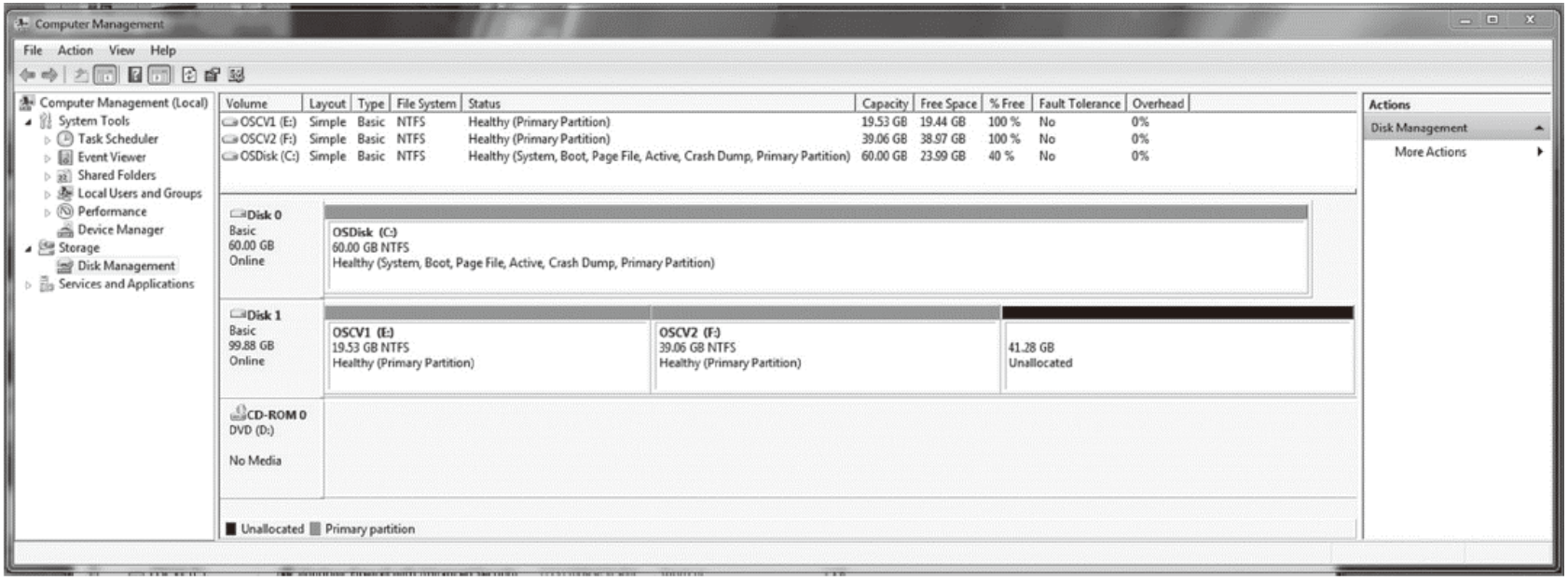
11.5 儲存裝置管理

- 當裝置被作業系統識別時，將讀取其分割資料，然後作業系統將為分割建立裝置項目 (在 Linux 中為 /dev)
- 從那時開始，一個組態檔案 (例如 /etc/fstab) 告訴作業系統在指定位置安裝包含檔案系統的每個分割，並使用如唯讀的安裝選項
- **安裝** (Mounting) 檔案系統使檔案系統可供系統及其使用者使用

11.5 儲存裝置管理

- 第二步是卷區的建立和管理
  - 有時候這個步驟是隱藏式的，就像將檔案系統直接放置在分割中一樣，然後可以安裝和使用該**卷區** (volume)
  - 在其它時間，卷區建立取消是顯示的
- 第三個步驟稱為**邏輯格式化** (logical formatting)，或稱為建立一個檔案系統
  - 在這個步驟中，作業系統將起始的檔案系統資料結構儲存到磁碟之中
  - 這些資料結構可能包含未使用和已配置空間的配置圖，以及一個起始的空目錄

圖 11.9 Windows 7 Disk Management 工具顯示裝置、分割、儲存空間和檔案系統



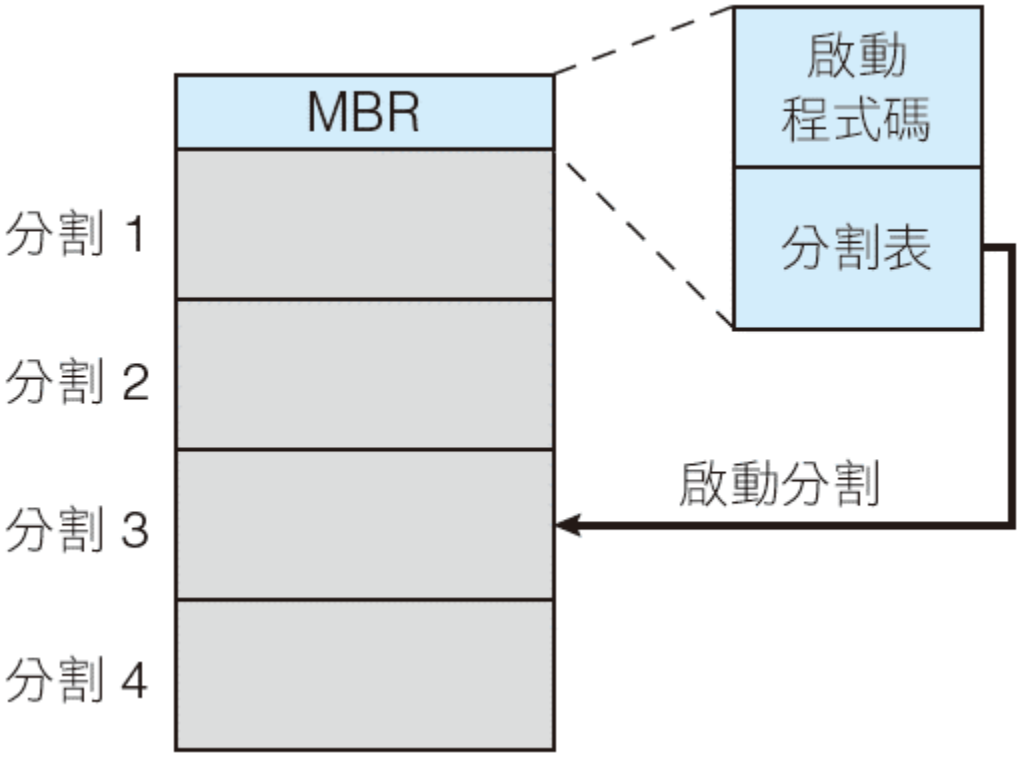
11.5 儲存裝置管理

- 為了增加效率，大部份檔案系統將區塊聚集成較大的區塊，常稱為叢集 (cluster)
  - 裝置 I/O 經由區塊完成，但是檔案系統 I/O 經由叢集完成，如此可以有效確保 I/O 有較多循序存取和較少隨機存取的性質
    - 例如，檔案系統嘗試在元資料附近增加檔案內容，減少對檔案進行操作時 HDD 磁頭的搜尋

11.5.2 啟動區塊

- 為了使電腦開始運作—例如當打開電源或重新啟動—必須有一個起始程式才能運作
  - 這個起始的**靴帶式** (bootstrap) 載入器往往相對簡單
  - 對於大多數的電腦而言，靴帶式載入器儲存在系統主機板的 NVM 快閃記憶體中，並映射到已知的記憶體中位置
  - 產品製造商可以根據需要進行更新，也可能被病毒感染，從而感染系統
  - 它初始化所有方面系統，從 CPU 暫存器到裝置控制器及主記憶體的內容

圖 11.10　Windows 中由儲存裝置啟動



11.5.3 毀損區塊

- 在關於毀損區塊的恢復方面，更複雜的磁碟系統更加聰明了
  - 控制器保有磁碟之中的毀損區塊串列
  - 毀損區塊串列起始於在製造工廠的低階格式化期間，而且在磁碟的整個使用壽命中都要更新資料
  - 低階格式化也設置一些作業系統無法看到的額外備份磁區
    - 並且在邏輯上可以告訴控制器用備份磁區置換每個毀損磁區，這個設計就是大家知道的**磁區備份** (sector sparing) 或**磁區轉換** (sector forwarding)

11.6 置換空間管理

- **置換空間管理** (swap-space management) 是作業系統的另一種低階工作
  - 虛擬記憶體需要使用輔助儲存器作為主記憶體的延伸
  - 既然裝置存取比記憶體存取慢很多，因此使用置換空間對於系統效能會有很大的降低
  - 置換空間的設計和實作的主要目的在於提供虛擬記憶體系統的最佳產量

11.6.1 置換空間使用

- 不同的作業系統會根據自己使用的記憶體管理方法，以不同的方式使用置換空間
  - 例如，使用置換方法的系統可能使用置換空間來保存整個行程的映像，包括
    - 程式碼
    - 資料區塊
  - 分頁系統可能只要儲存被移出主記憶體的分頁
  - 一個系統的置換空間需要從幾百萬位元組到幾十億位元組的磁碟空間
    - 是根據它在備份時所需的實體記憶體和虛擬記憶體數量，以及虛擬記憶體使用的方式而不同

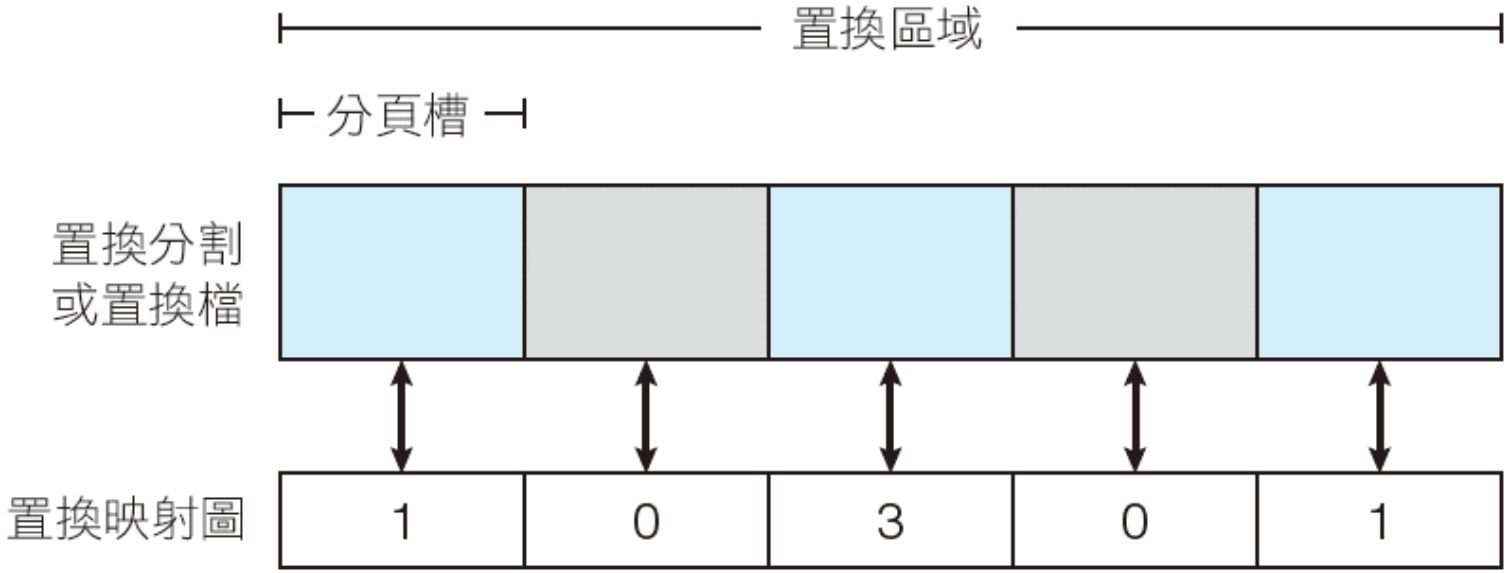
11.6.1 置換空間使用

- 超估置換空間要比低估安全，因為一個系統如果置換空間不足夠時，可能被中止行程或完全毀損
  - 超估造成本來可以給檔案使用的磁碟空間浪費，但沒有其它傷害。
  - 一些系統推薦對置換空間設置額外的數量，例如
    - Solaris 建議設置置換空間等於虛擬記憶體超過可分頁實體記憶體的數量
  - 在過去，Linux 建議設置置換空間為實體記憶體數量的兩倍
    - 現在這項限制已經消失，且大部份 Linux 系統使用相當少的置換空間

11.6.1 置換空間使用

- 有些作業系統——包含 Linux ——允許使用數個置換空間，包含
  - 檔案
  - 特定的置換分割
- 這些置換空間通常放在不同的儲存裝置上，因此分頁和置換造成的 I/O 系統負擔可以平均地分散到系統的 I/O 頻寬

圖 11.11　Linux 系統中置換的資料結構



11.7.1 主機附加儲存

- 高級工作站和伺服器通常需要更多的儲存或需求共享儲存，因此使用更複雜的 I/O 架構
  - 例如**光纖通道** (fibre channel, FC)，可以超越在光纖上運作的高速串接結構或四芯電纜
  - 由於位址空間大，多個主機和儲存裝置的置換性質可以連接到結構
  - 從而在 I/O 通信中提供極大的靈活性

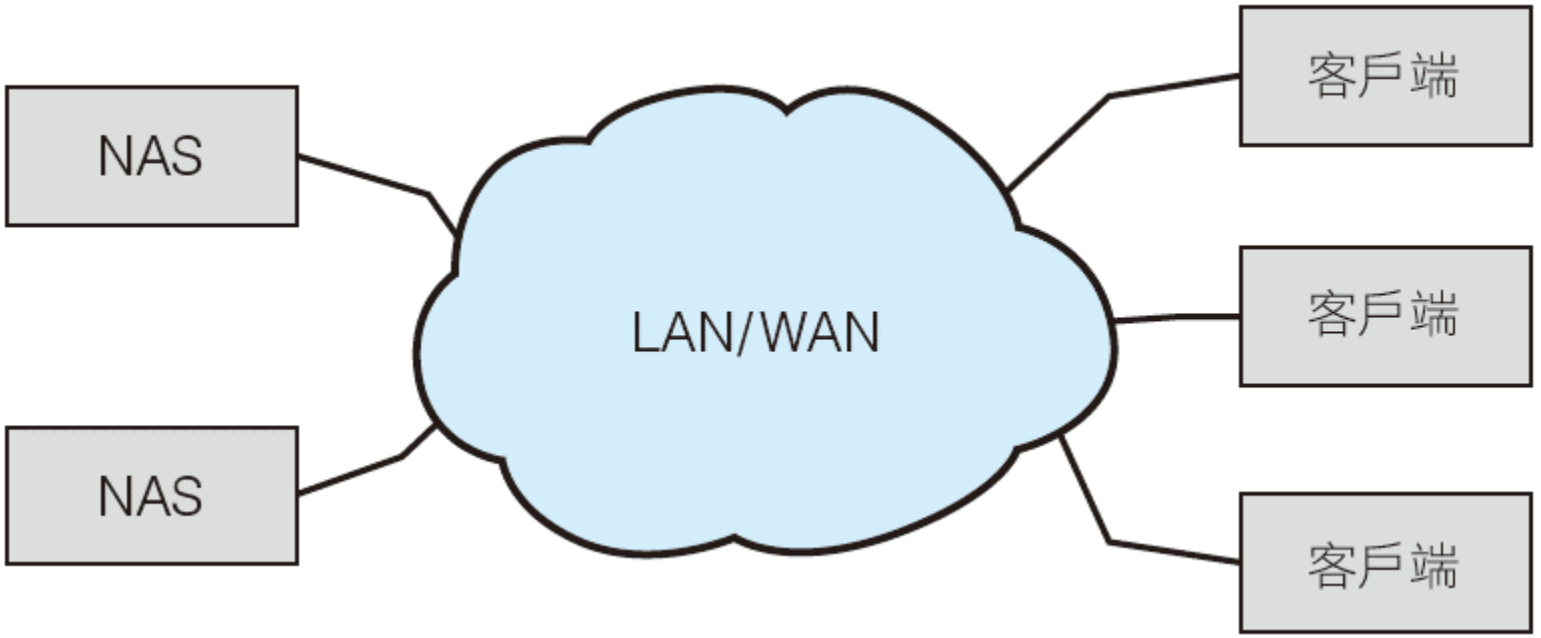
11.7.1 主機附加儲存

- 各式各樣的儲存裝置都適合使用作主機附加儲存，其中包括
  - HDD；NVM 裝置；CD、DVD、藍光和磁帶裝置和**儲存區域網路** (storage-area networks, SAN)
  - 啟動向主機附加儲存裝置的資料傳輸的 I/O 命令
    - 讀取和寫入定向到專門標識儲存單元的邏輯資料區塊
    - 例如匯流排 ID 或目標邏輯單元

11.7.2 網路附加儲存

- **網路附加儲存** (network-attached storage, NAS) 是經由網路存取儲存的方式
  - 客戶經由遠端程式呼叫的介面存取網路附加儲存
  - 遠端程序呼叫 (remote procedure calls, RPC) 在 IP 網路上經由TCP 或 UDP 上執行
    - 通常 IP 網路是對客戶端執行所有資料傳送的同一個區域網路
    - 網路附加儲存單元通常是使用 RPC 介面的軟體實作成儲存陣列

圖 11.12　網路附加儲存



11.7.2 網路附加儲存

- CIFS 和 NFS 提供各種鎖定功能，允許使用這些協定在存取 NAS 的主機之間共用檔案
  - 例如，登錄多個 NAS 客戶端的使用者可同時從所有這些客戶端存取其主目錄
- 網路附加儲存對於所有在 LAN 上的電腦提供共用一群儲存裝置的方法，並且享有和區域主機附加儲存相同的命名和存取方便
  - 這種方式比一些直接附加儲存的選項沒有效率，而且效能較差

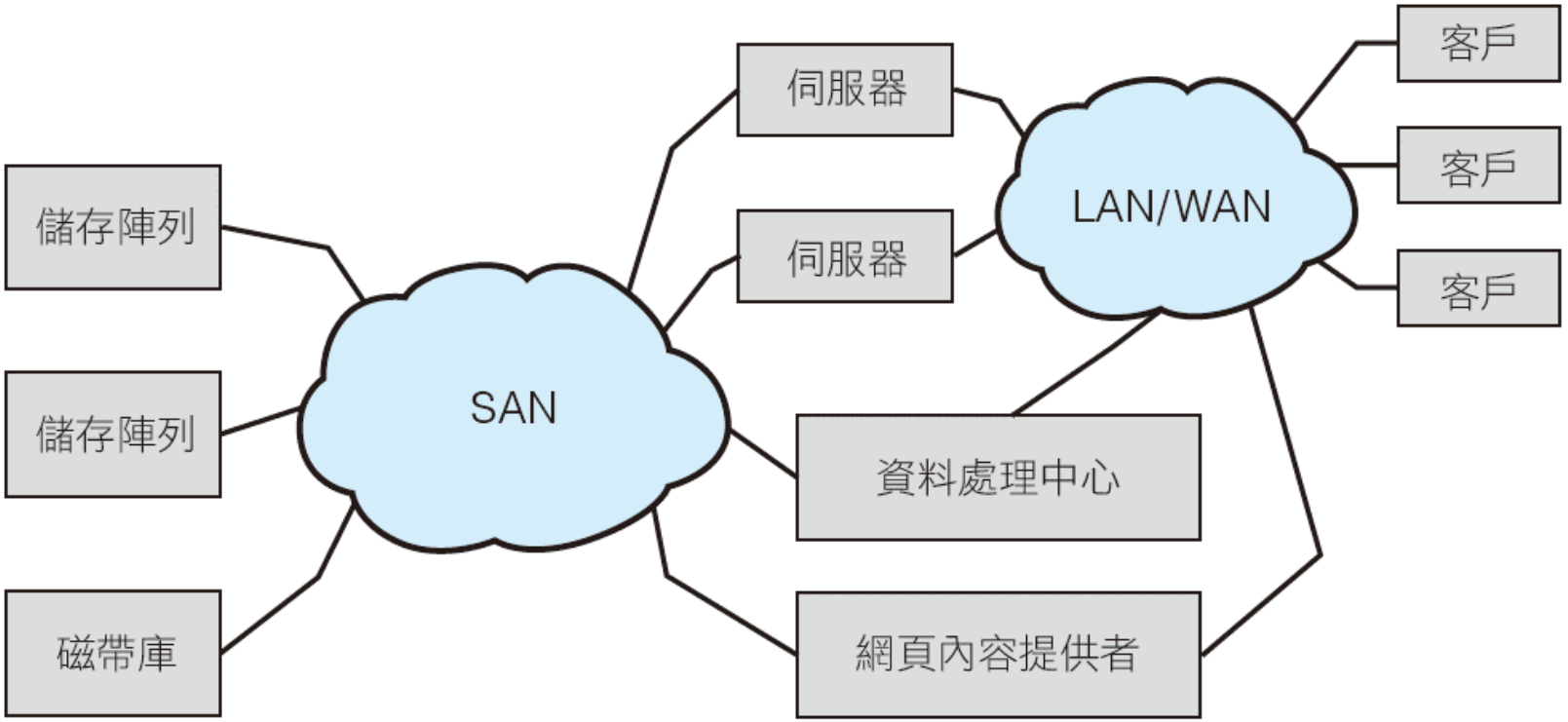
11.7.2 網路附加儲存

- **iSCSI** 是最近的網路附加儲存協定
  - iSCSI 使用網路 IP 協定來實現 SCSI 協定
  - 在主機和它們的儲存體之間互相連接的是網路
    - 非 SCSI 線路
    - 主機可以將其儲存方式視為直接附加，即使儲存體與主機之間的距離較遠
    - NFS 和 CIFS 提供一個檔案系統經由網路存取檔案
    - iSCSI 則是藉由網路傳送邏輯區塊，並讓客戶可以直接使用區塊或建立檔案

11.7.4 儲存器區域網路和儲存陣列

- 網路附加儲存器系統的一項缺點是，儲存裝置的 I/O 運作占用網路的頻寬，因此增加網路通信的潛伏時間
  - 這個問題在大型客戶端—伺服器的安裝上特別顯著—伺服器與客戶端間的通信以及伺服器和儲存裝置間的通信爭取頻寬
- **儲存器區域網路** (storage-area network, SAN) 是伺服器和儲存器單元間的私人網路，如圖 11.13 所示
  - SAN 的功能在於它的彈性
  - 許多主機和許多儲存陣列可以連接到相同的 SAN，而儲存器可以動態地分配給主機

圖 11.13　儲存器區域網路



11.7.4 儲存器區域網路和儲存陣列

- 儲存陣列可以採用 RAID 或無保護的磁碟機 [僅一組磁碟 (Just a Bunch of Disks, JBOD)]
- 一個 SAN 開關允許或禁止主機與儲存器之間的存取
  - 舉一個例子，如果主機的磁碟空間降低時，SAN 可以分配更多儲存器給該主機
  - SAN 使伺服器的叢集共用相同儲存器，以及儲存陣列多重直接主機連接變為可能
  - 通常 SAN 比儲存陣列有較多的埠——且較不昂貴埠
  - SAN 連接是短距離的，並且通常沒有路由，因此 NAS 可以擁有比 SAN 來得多的連接主機。

11.7.4 儲存器區域網路和儲存陣列

- 儲存陣列是一種專用裝置 (見圖 11.14)
  - 包括 SAN 埠和/或網路埠
  - 還包含用於儲存資料的裝置和一個控制器，來管理儲存並允許透過網路儲存存取
  - 控制器由 CPU、記憶體和使用軟體來實作陣列功能，這些功能可以包括
    - 網路協定、使用者介面、RAID 保護、快照、副本、壓縮、重複資料刪除和加密

圖 11.14　一個儲存陣列

11.7.4 儲存器區域網路和儲存陣列

- 一些儲存陣列包括 SSD
  - 陣列可以僅包 SSD，從而獲得最大的效能，但容量較小，或者可以包含 SSD 和 HDD 的混合使用
  - 陣列軟體 (或管理員) 選擇給定用途或將 SSD用做快取，並將 HDD 批次使用的最佳介質儲存
- FC 是最普通 SAN 連接，雖然 iSCSI 的簡單讓它的使用增加

11.8 RAID 結構

- 各種磁碟組織的技巧，統稱為**不昂貴磁碟的重複陣列** (redundant array of independent disk, RAID)，常被用來強調效能和可靠度
  - 在過去
    - RAID 是由一些小型便宜的磁碟機組成，它被視為是大型、昂貴磁碟的經濟替代品
  - 今日
    - RAID 被使用則是因為它們的高可靠度和較高的資料傳送速率，而非因為經濟因素
  - RAID 中的 I 表示“獨立”(independent)，而非“不昂貴”(inexpensive

11.8.1 經由重複改進可靠度

- HDD 的 RAID 可靠度
  - *N* 台磁碟機的磁碟機組中某台磁碟機失效的機率比一台特定磁碟機失效的機率還高
  - 假設單一台磁碟機的**平均失效時間** (mean time to failure, MTBT) 是 100,000 小時，則 100 台磁碟機的磁碟陣列中，某台磁碟機的平均失效時間是 100,000/100 = 1,000 小時 (或 41.66 天)，這不是很長的時間！
  - 如果我們只儲存這些資料的一份複製，則每台磁碟機的失效將造成可觀的資料遺失
    - 這種高比率的資料遺失是無法接受的

11.8.1 經由重複改進可靠度

- 一台鏡射磁碟機的平均失效間—在這裡失效是指資料遺失—取決於兩項因素：
  - 個別磁碟機的平均失效時間
  - **平均修復時間** (mean time to repair)
    - 平均修復時間換掉失效的硬碟，並且恢復其上的資料所花費的時間
    - 假設兩台磁碟機的失效是彼此獨立 (independent)
      - 一台磁碟機失效不會連結到另一台磁碟機
      - 如果單一台磁碟機的平均失效時間是 100,000 小時，而且平均修復時間是 10 小時，則一個鏡射系統的**平均資料遺失時間** (themean time to data loss) 是 $100{,}000^2 / (2 * 10) = 500 * 10^6$ 小時，即 57,000 年！

11.8.1 經由重複改進可靠度

- 電源失效是另一個不安的來源，因為其發生的機會遠比自然災害頻繁
  - 即使使用鏡射磁碟機，如果寫入到兩台磁碟機的相同區段正在進行時，在這兩個區塊完全寫入前電源失效的話，則這兩個區塊都會處於不一致的狀態
  - 這個問題的解決方法是先寫入一份複製，然後再寫入第二份複製，因此這兩份複製中的一份一定是前後一致
  - 另一種是加非揮發性 RAM (nonvolatile RAM, NVRAM) 快取到 RAID 陣列，這種寫入備份快取電源失效時，保

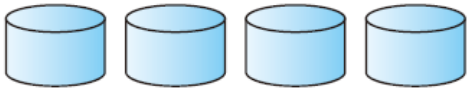
11.8.2 藉由平行改善效能

- 有了多台磁碟機時，我們也可以藉由在多台磁碟間分割儲存資料來改進傳輸速率
  - **資料分割** (data striping) 的最簡單格式是，將每一個位元組分成位元，分散在許多台磁碟機上
  - 這種分割稱為**位元層次的分割** (bit-level striping)

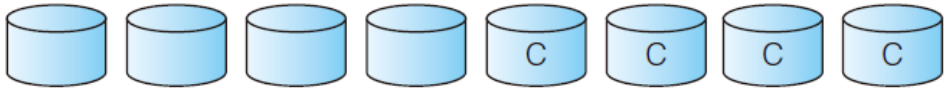
11.8.3 RAID 層次

- 鏡射提供高的可靠度，但是卻很昂貴
- 分割提供高的資料傳輸率，但是卻不能增進可靠度
- 數種藉由使用磁碟機分割結合“同位”位元的低價格重複性技巧曾被提出，這些技巧有不同的價格
  - 效能調整，而且被歸納成稱為 **RAID 層次** (RAID level) 的數種階層
- 圖 11.15 是以圖型的方式顯示
  - 圖中的所有描繪情況，相當於四台磁碟機的資料被儲存，而多餘的磁碟機則用來儲存錯誤發生時復原用的重複資料

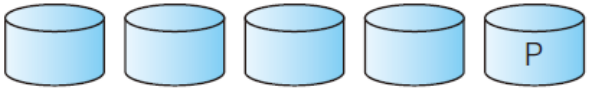
圖 11.15 RAID 層次



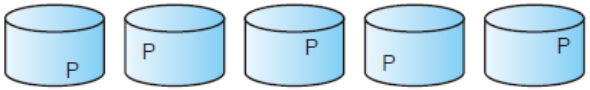
(a) RAID 0：非重複分割



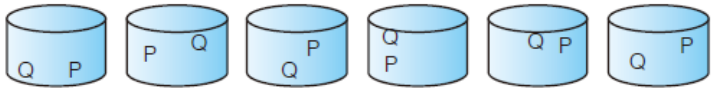
(b) RAID 1：鏡像磁碟



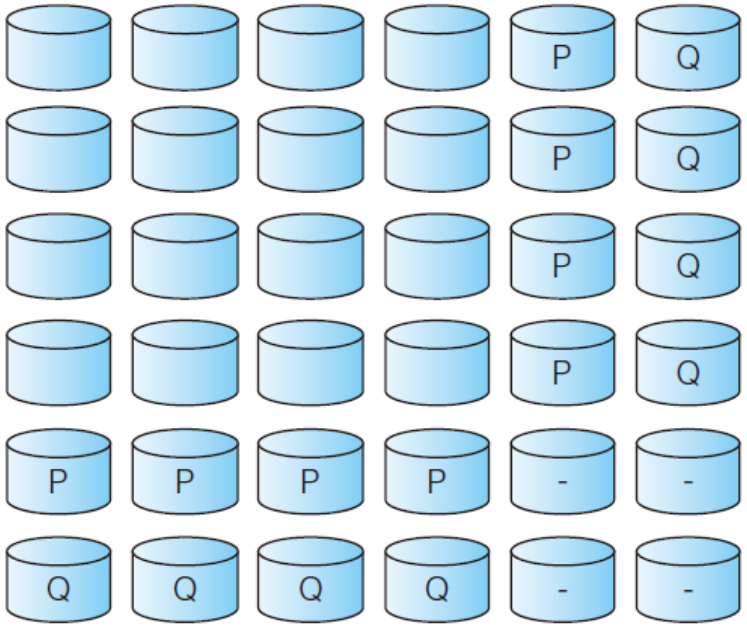
(c) RAID 4：區塊交錯的同位位元



(d) RAID 5：區塊交錯的分散式同位位元



(e) RAID 6：P + Q 冗餘



(f) 多維 RAID 6

11.8.3 RAID 層次

- RAID 層次 0：
  - RAID 層次 0 是指在區塊層次分割的儲存陣列，但是沒有任何重複的資料 (例如鏡射或同位位元)，如圖 11.15(a) 所示
- RAID 層次 1：
  - RAID 層次 1 是指磁碟的鏡射。圖 11.15(b) 顯示一個鏡射組織
- RAID 層次 4：
  - RAID 層次 4，也稱為記憶體形式的錯誤更正碼組織 [asmemory-style error-correctingcode (ECC) organization]
  - ECC 也被使用於 RAID 5 和 6

11.8.3 RAID 層次

- RAID 層次 5：
  - RAID 層次 5 亦即區塊分割分散式，藉由把資料和同位位元分散到 $N+1$ 台磁碟機，這和層次 4 的儲存資料在 N 台磁碟機，而同位位元在某台磁碟機不相同
  - 對於區塊 N 而言，其中一台儲存同位位元，而其餘的儲存資料
    - 例如，五台磁碟機的陣列，第 n 個區塊的同位位元儲存在磁碟機 $(n \bmod 5)+1$；其它四台磁碟機的第 n 個區塊儲存該區塊的實際資料
    - 同位位元區塊不能儲存同一台磁碟機區塊的同位位元，因為磁碟毀損將造成資料和同位位元一起毀損，因此將無法復原

11.8.3 RAID 層次

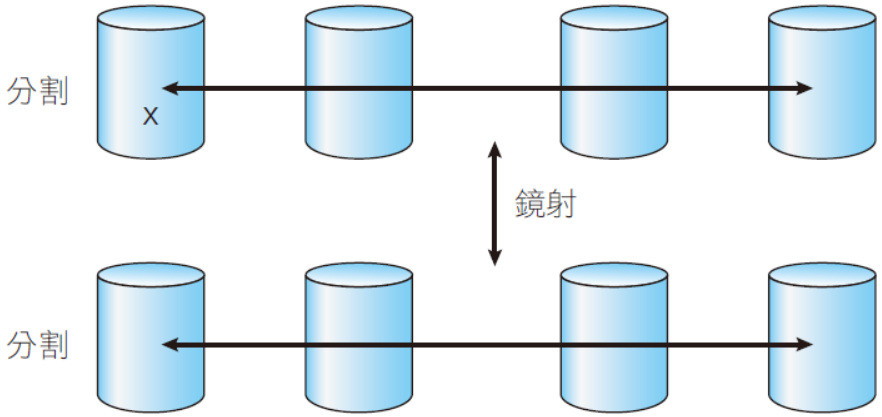
- 藉由散佈同位位元到同一組中的所有磁碟機，RAID 5 避免對單一台同位位元磁碟機的過度使用，而這可能發生在 RAID 4
- RAID 5 是最普遍的同位位元RAID 系統

- RAID 層次 6：
  - RAID 層次 6 也稱為 **P + Q 重複技巧** (P + Q redundancy scheme) 和RAID 層次 5 極相似，但是儲存額外重複性資料，以保護多台磁碟機失效
  - XOR 同位不能在兩個同位區塊上使用，因為它們是相同的，並不會提供更多修正訊息
  - 不使用同位位元，而使用錯誤更正碼 [例如有限域數學 (Ealois field math)] 來計算 Q

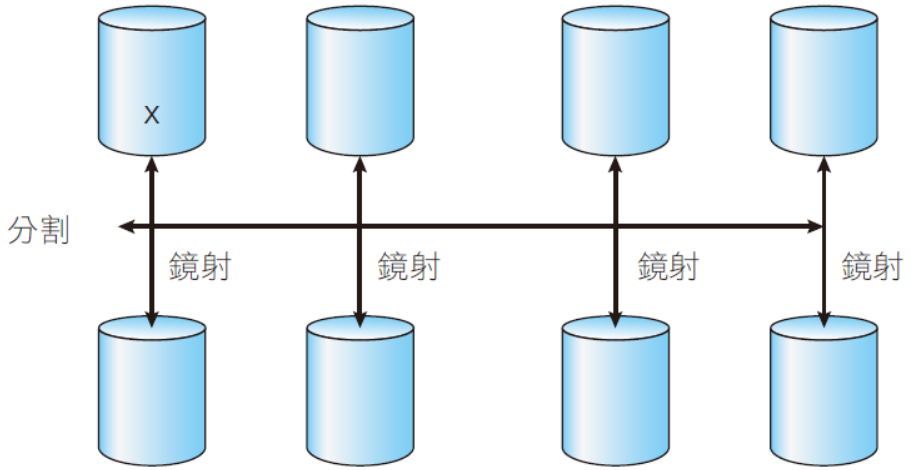
11.8.3 RAID 層次

- 在圖11.15(e) 所顯示的技巧中，每 4 個位元的資料就有 2 位元的額外資料被儲存
  - 這和層次 5 中的 1 個同位位元不相同
  - 此系統可以忍受兩台磁碟機壞掉

11.16 單一磁碟機失效的 RAID 0 + 1 和 1 + 0



(a) 單一磁碟機失效的 RAID 0 + 1



(b) 單一磁碟機失效的 RAID 1 + 0

11.8.3 RAID 層次

- **快照** (snapshot) 是上次更新發生前檔案系統的景象
- **複製** (replication) 牽涉到在重複和災害復原個別位置間寫入的自動複製
  - 可以是同步或非同步
    - 在同步複製，每個區塊在完全寫入之前必須局部和遠端地寫入
    - 在非同步複製，寫入內容聚集在一起且定期地寫入
  - 如果主要位置失敗，非同步複製可能產生資料遺失，但是非同步複製較快且沒有距離限制
  - 複製越來越多地使用於資料中心，甚至是主機中

11.8.3 RAID 層次

- 作為 RAID 保護的替代方案，複製可防止資料遺失並提高讀取效能
- 當然可以使用更多比大多數類型的 RAID 儲存
- 這些特性的實作取決於 RAID 製作的階層而有所不同
  - 例如，RAID 以軟體製作，則每個主機必須製作和管理自己的複製
  - 如果 RAID 在儲存陣列製作或在 SAN 連接製作，主機資料可以被複製

11.8.3 RAID 層次

- 大部份 RAID 實作上的另一個層面，是熱備份磁碟或磁碟機組
  - 一台**熱備份** (hot spare) 的磁碟機並不是用來處理資料，而是被架構成如果有其它磁碟機壞了，可以用來做為替代品
    - 例如，如果鏡像組中的一台磁碟機壞了時，熱備份可以用來重建鏡像組
    - 依照這種方法，RAID 層次可以自動地重建，而不需要等到壞掉的磁碟機被替換
    - 配置一台以上的熱備份，可以讓超過一個以上的失效不用人力介入就自動被修復

11.8.6 RAID 的問題

- RAID 不永遠保證資料是可以使用的
  - 舉例來說，檔案的一個指標可能是錯誤的，或檔案結構裡面的指標可能是錯誤的
  - 不完全的寫入 [稱為不完全寫入 (torn writes)] 如果不適當地恢復，會造成錯誤百出的資料
  - 有些其它的行程也可能偶然在檔案系統的結構上寫入
  - RAID 保護實體媒介的錯誤，但不是其它硬體和軟體的錯誤
  - 軟體和硬體的錯誤一樣大，在系統的資料上也就有多少潛在危險

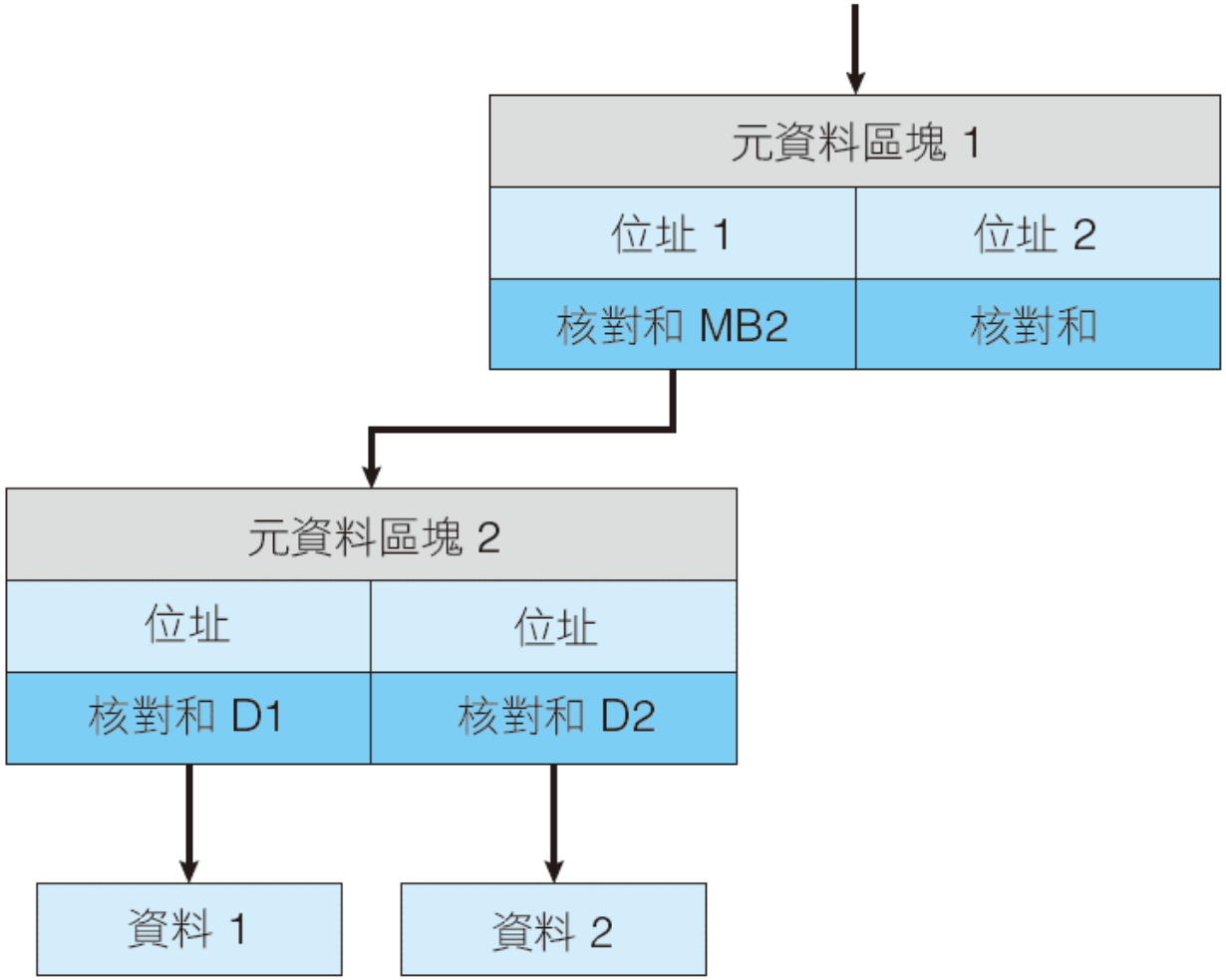
11.8.6 RAID 的問題

- Solaris ZFS 檔案系統採取的創新方式，經由檢查碼 (checksum) 來解決這些問題
  - 檢查碼是一種資料完整性的技術
  - ZFS 維持所有區塊內在的檢查碼，包括資料和元資料
  - 檢查碼不與正在被檢查的區塊放在一起，而是與指標一起儲存在區塊中 (見圖 11.17)
  - 考慮 **inode**—儲存檔案系統元資料的一種資料結構—是指向它的資料一個指標
  - 在inode 裡面是每個資料區塊的檢查碼
  - 如果資料有問題，檢查碼將是不正確的，而且檔案系統將知道檢查碼不正確

11.8.6 RAID 的問題

- 如果資料被鏡射，並且有一個正確檢查碼的區塊和一個不正確檢查碼區塊，ZFS 將自動地用一個好的區段更新壞的區段
- 指向 inode 的目錄進入點對 inode 有一個檢查碼
- 當目錄被存取時，會發現在 inode 的任何問題
- 檢查碼發生遍及所有的 ZFS 結構，比 RAID 磁碟組或標準檔案系統提供更高水準的一致性、錯誤發現和錯誤更正
- 因為 ZFS 的整體效能是很快速的，所以檢查碼計算的額外負擔和讀取—修正—寫入循環產生的額外區塊不會引人注目

圖 11.17　ZFS 檢查所有元資料和資料的檢查碼



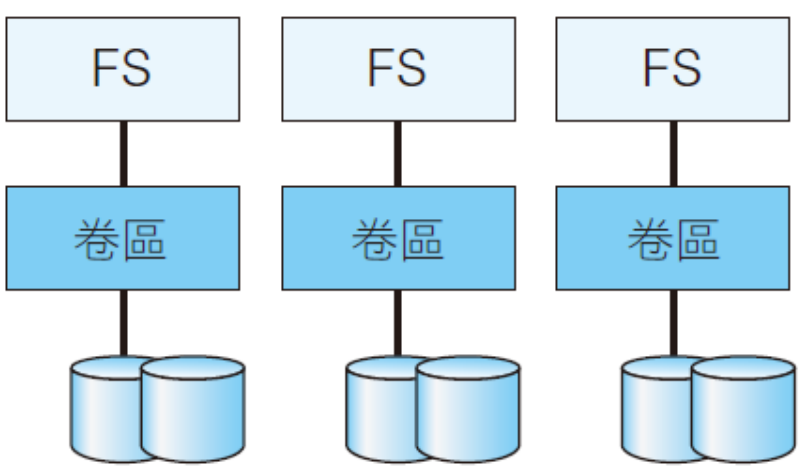
11.8.6 RAID 的問題

- ZFS 結合檔案系統管理和卷區管理成為一個單元，提供比傳統允許這些功能分離更大的功能性
  - 磁碟或磁碟的分割區經由 RAID 集結成一個儲存**池** (pool)
  - 一個儲存池可以保有一個或多個 ZFS 檔案系統，整個池皆為使用空間可以被在這個池的所有檔案系統使用
  - ZFS 使用 malloc() 和 free() 的記憶體模式，當區塊在檔案系統內被使用或釋放時，為每個檔案系統配置或釋放記憶體

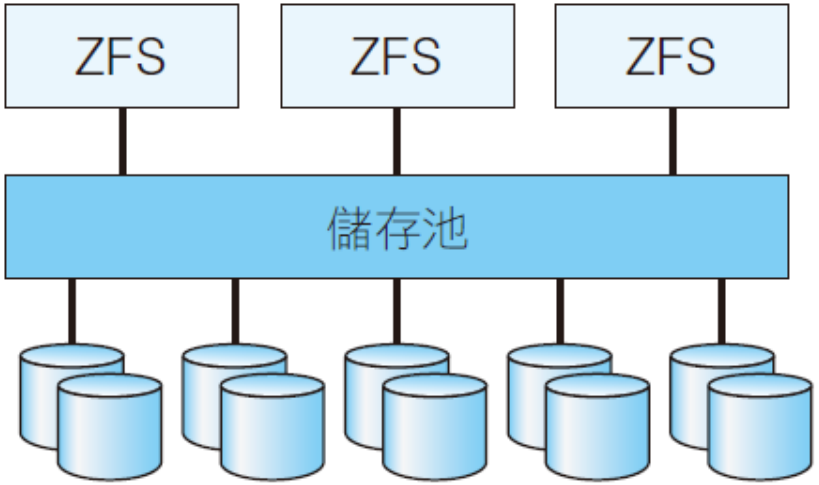
11.8.6 RAID 的問題

- 在儲存器的使用上沒有人為的限制，並且不需要在卷區重新配置檔案系統，或是重新改變卷區的大小
- ZFS 提供配額以限制一個檔案系統和保留區的大小，以確保檔案系統可用特定的數量成長，但這些變數可以被檔案系統擁有者在任何時候改變
- 其它系統 (例如 Linux) 具有卷區管理器，該卷區管理器允許有邏輯的連接多個硬碟以建立大於磁碟的卷區來容納大型檔案系統
- 圖 11.18(a) 描繪傳統的卷區和檔案系統，而 11.18(b) 則顯示 ZFS 模型

圖 11.18　ZFS 模型的傳統卷區和檔案系統



(a) 傳統的卷區和檔案系統



(b) ZFS 和儲存池