
ASIA EDITION
作業系統
Operating System Concepts
TENTH EDITION
趙涵捷 審閱
吳庭育 駱詩軒 譯
ABRAHAM SILBERSCHATZ
PETER BAER GALVIN
GREG GAGNE
東華書局
WILEY

Chapter 1

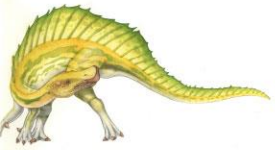
概　說

章節目標

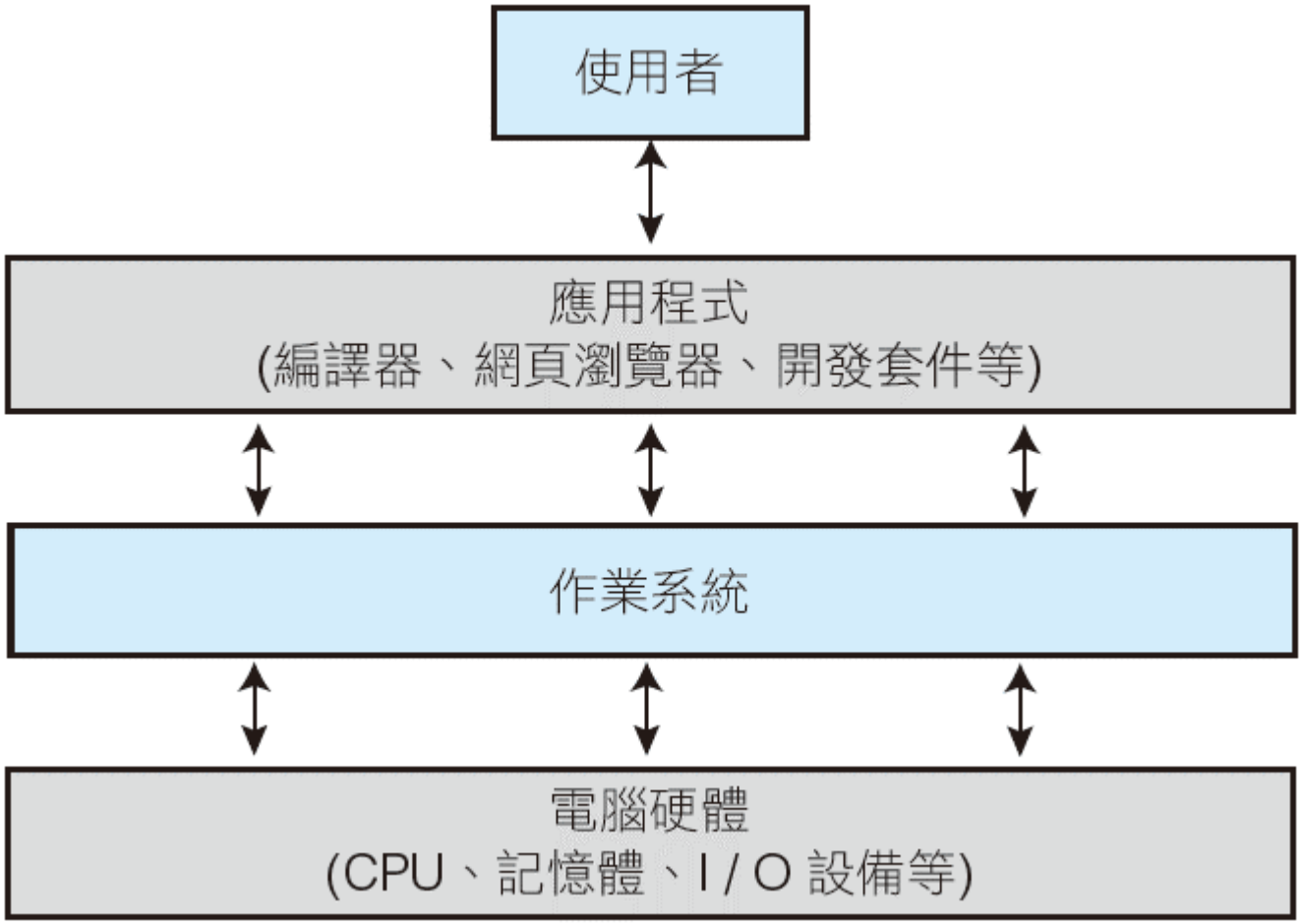
- 描述電腦系統的基本組織和中斷扮演的角色
- 描述現代多處理器電腦系統中的各項元件
- 圖形闡釋從使用者模式到核心模式的轉換
- 探討作業系統如何被使用在各種計算環境中
- 提供免費及開放原始碼作業系統之範例

作業系統做什麼

- 我們由檢視作業系統在整個電腦系統中的角色開始討論起
- 一部電腦系統大致上可以分為四個單元：
  - 硬體
  - 作業系統
  - 應用程式
  - 使用者



電腦系統各項組件關係示意圖



作業系統做什麼

- **硬體** (hardware)
  - **中央處理器** (central processing unit, CPU)、**記憶體** (memory)、**輸入/輸出裝置** [input/output (I/O) device]
    - 提供系統基本的運算資源
  - **應用程式** (application program)
    - 文書處理程式、試算表、編譯器和網頁瀏覽器
      - 藉著有效運用這些資源，來解決使用者的運算問題
  - 作業系統控制硬體，並針對不同使用者協調其在各種應用程式之間的使用

使用者觀點

- 在這種情況下，作業系統主要是設計成**容易使用** (ease of use)，有些注意力是花在性能和安全性上，對於**資源的使用** (resouce utilization)
  - 不同硬體和軟體資源如何被分享則不必費心
- 有些電腦只有很少或完全沒有使用者的觀點
  - 例如，家用設備**嵌入式電腦**

系統觀點

- 視作業系統為一個資源分配者 (resource allocator)
  - 作業系統就像是這些資源的管理者，面對眾多且可能衝突的資源要求，作業系統必須決定如何分配這些資源給特定的程式與使用者，以便讓它有效率且公平地操作電腦
  - 對於作業系統的另一種稍微不同的觀點是強調，從控制不同的 I/O 裝置與使用者程式的需要上來看
  - 作業系統是一套控制程式，這一套**控制程式** (control program) 是用來管理使用者程式的執行以防止錯誤產生，及對電腦的不正確使用

定義作業系統 (1/2)

- 完全適切的作業系統定義
  - 常見的定義，而且也是我們經常遵守的定義是，作業系統是一個在電腦內部隨時都在執行的程式
    - 一般稱為**核心** (kernel)
  - 伴隨著核心，有兩個其它型態的程式：
    - **系統程式** (system program)
    - 應用程式

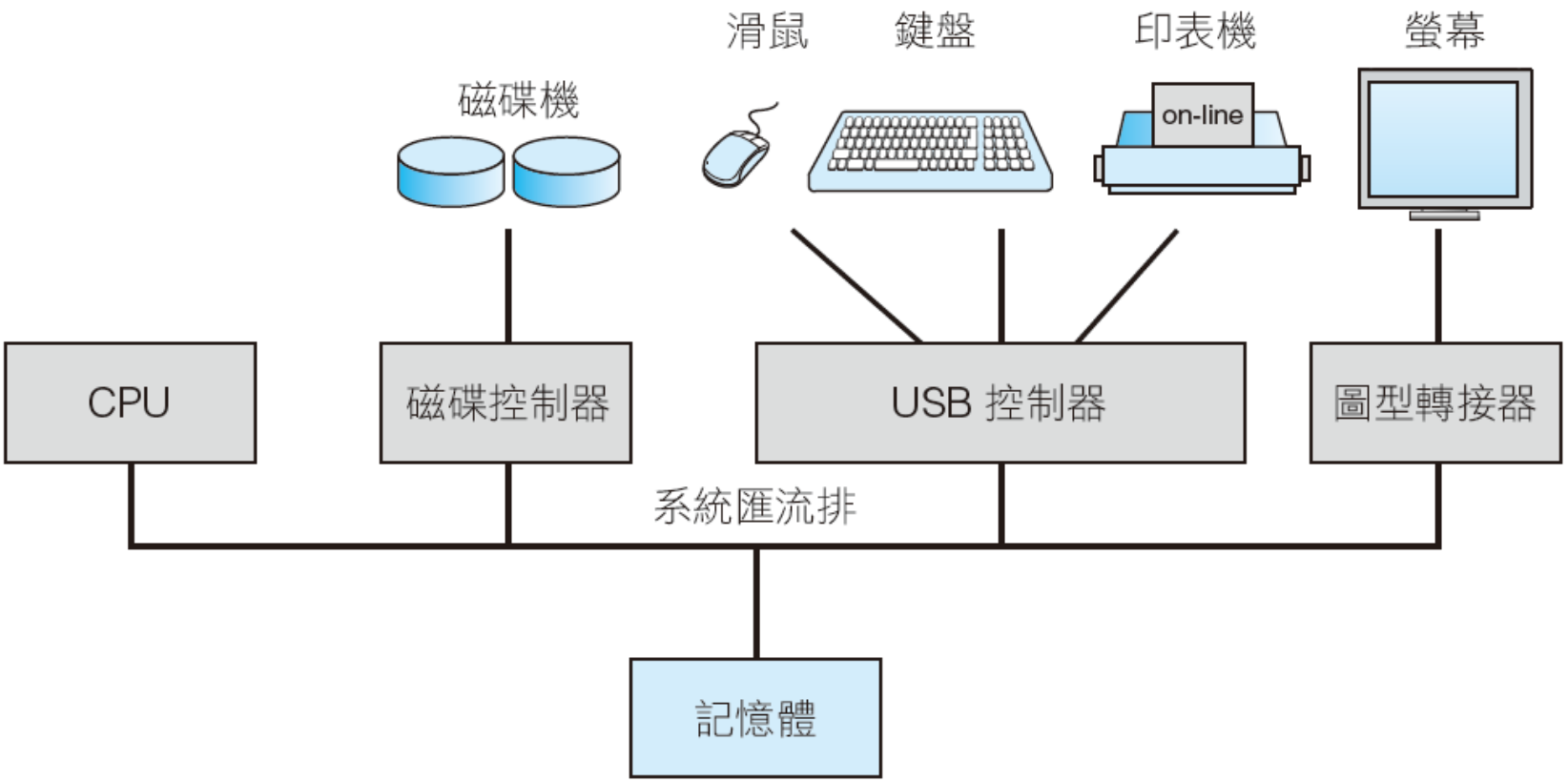
定義作業系統 (2/2)

- 一旦核心載入完成後，便可以開始向系統及其使用者提供服務
  - 有些服務是由系統程式在核心外部來提供，這些程式載入到記憶體中成為**系統守護進程** (system daemons)，這些守護程序在核心中會在整個過程中一直運行將持續等待事件的請求

電腦系統組織

- 近代的一般用途電腦系統包含一個或更多 CPU 和一些裝置控制器
  - 它們經由提供存取共用記憶體的公用匯流排 (bus) 連接在一起
- CPU 和裝置控制器可以同時執行，互相競爭記憶體週期

近代電腦系統



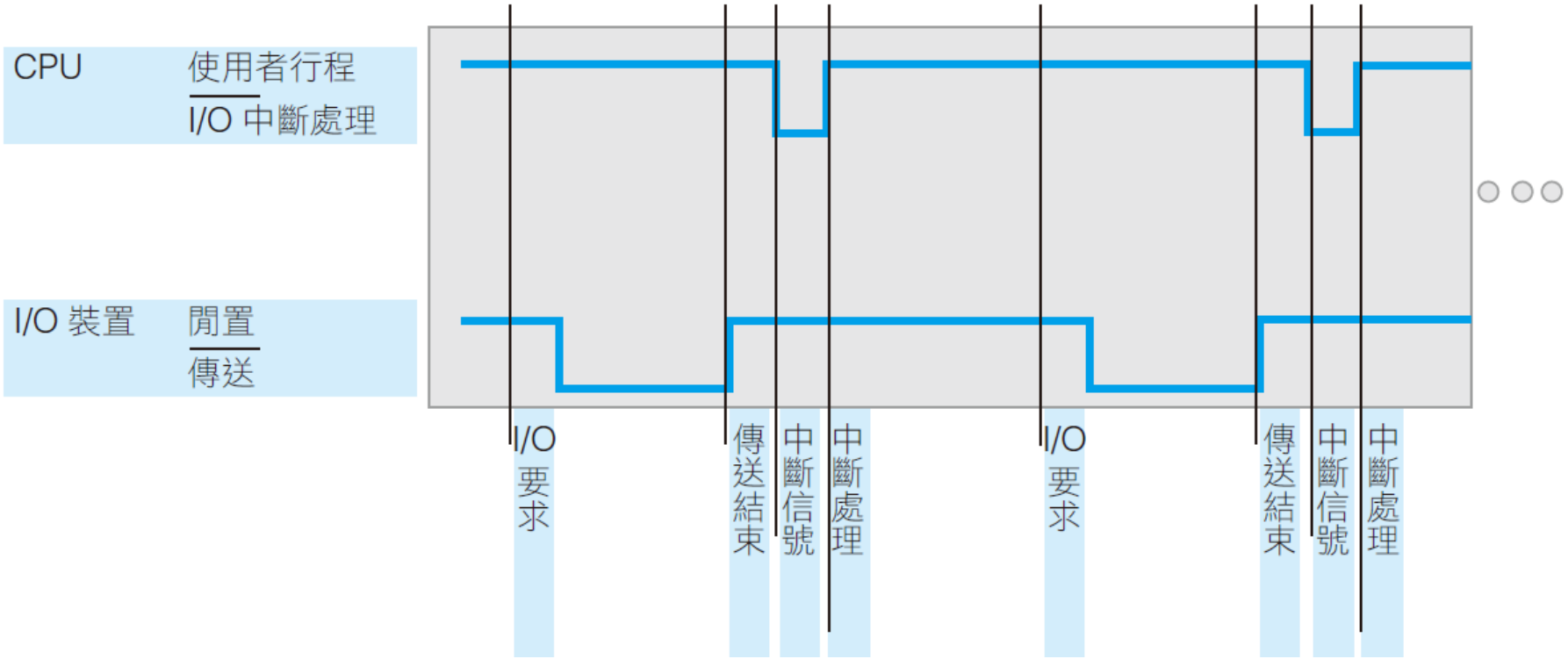
中斷 (1/2)

- 對於傳統的輸入/輸出的計算機的運作方式
  - 輸入/輸出開始執行前，驅動程式會先載入到裝置控制器中適合的暫存器中
    - 然後裝置控制器會確認這些暫存器內的內容來決定後續要進行的動作
      - 例如：“從鍵盤讀取字元”
  - 裝置控制器開始將資料從設備上傳輸到本地端緩衝區
  - 當資料傳輸完成後，裝置控制器將會通知設備驅動程式其操作已經完成
- 這是動作將藉由**中斷** (interrupt) 來完成

中斷(2/2)

- 陣列 [或叫作**中斷向量** (interrupt vector)]
  - 是以特定號碼當作索引值，只要有中斷要求，就提供產生中斷裝置的中斷服務常式之位址
- 中斷的架構也必須保存中斷狀態的訊息內容
  - 以便在中斷結束後可以恢復到中斷前的狀態
  - 中斷的另一種形式是**陷阱** (trap) [或**異常**(exception)]
    - 它是由軟體生成的中斷，起因於任一方的錯誤
      - 例如，被零除或無效的記憶體存取
    - 或使用者程式的特定請求可以藉由執行稱為**系統呼叫** (system call) 的特殊操作來執行系統服務

對執行輸出之單一行程的中斷時間線



儲存體的定義和記號

- 電腦儲存的基本單元是位元 (bit)。一個位元可以包含 0 和 1 這兩個數值中的一個
- 電腦的所有其它儲存單位是根據位元的集合。有足夠的位元，令人驚訝的是，一台電腦可以表示多少東西
  - 數字、文字、影像、電影、聲音、文件和程式
- 一個位元組(byte) 是 8 個位元，在大部份電腦，位元組是最小的儲存塊
  - 大部份電腦沒有搬移一個位元的指令，但都有搬移一個位元組的指令
- 另一個較少見的術語是字元組 (word)，這是一台電腦架構的基本資料單位。一個字元組是由一個或多個位元組所組成。
  - 一台有64 位元暫存器和 64 位元定址能力的電腦通常有 64 位元 (8 位元組) 的字元組。
  - 電腦執行許多指令時大多是以自己的字元組為單位，而非一次一個位元組

儲存體的定義和記號

- 電腦的儲存量與大部份電腦的處理能力，通常是以位元組或位元組的集合作為衡量和處理的單位
  - 1 KB (kilobyte) 是 1024 位元組
  - 1 MB (megabyte) 是 10242 位元組
  - 1 GB (gigabyte) 是 10243 位元組
  - 1 TB (terabyte) 是 10244 位元組
  - 1 PB (petabyte) 是 10245 位元組
- 電腦製造商通常都將這些數字四捨五入，並稱 1 MB 是 1 百萬位元組，1GB 是十億位元組
- 網路測量是以上通則的例外，它們是以位元為單位 (因為網路移動資料時一次一個位元)

儲存體結構

- CPU 只能從記憶體載入程式，所以任何被執行的程式都必須儲存在這裡
  - 一般用途電腦所執行的程式大部份是來自於可複寫的記憶體
  - 它們被稱為主記憶體
    - 又叫作隨機存取記憶體 (random-access memory, RAM)

儲存體結構

- 電腦也使用其它形式的記憶體
  - 例如，電腦開機時運行的第一個程式是靴帶式程式(bootstrap program)，接著載入作業系統
    - 由於 RAM 是**揮發性** (volatile) 記憶體，當關閉電源時儲存其中的內容將會消失
      - 因此我們無法使用它來儲存靴帶式程式
      - 取而代之的是電腦使用了
        - » 電子抹除式可複寫唯讀記憶體 (EEPROM)
        - » 其它形式的**韌體** (firmware)
      - 為一種價位較高的永久性記憶體

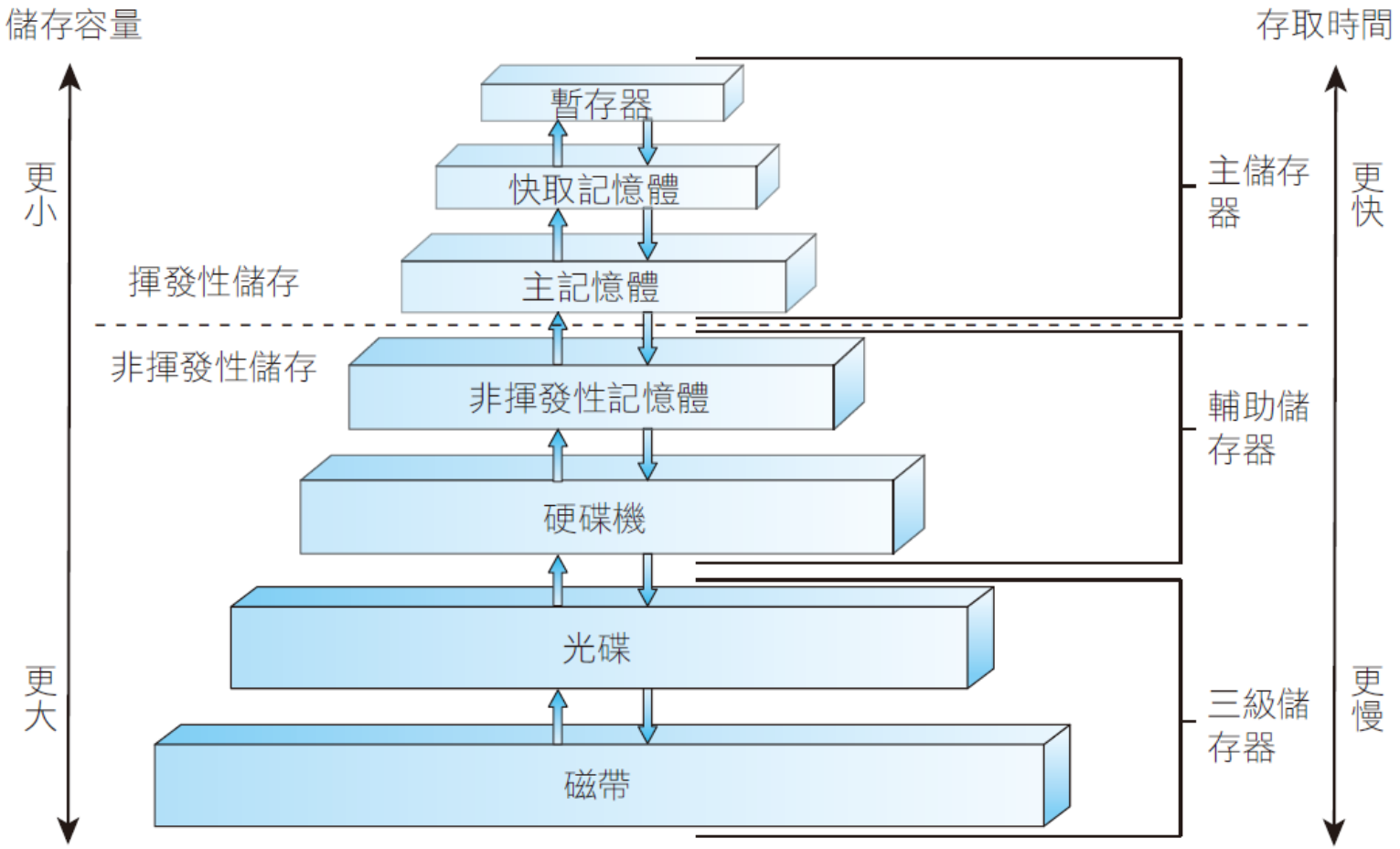
儲存體結構

- 大部份電腦系統提供輔助儲存器 (secondary storage) 作為主記憶體的延伸
  - 輔助儲存器的主要要求是能夠永久保存大量的資料
  - 最常見的輔助儲存裝置是
    - 硬碟機 (hard-disk drives, HDD)
    - 非揮發性記憶體設備[nonvolatile memory (NVM) devices]
  - 它可以提供程式和資料的儲存

儲存體結構

- 大部份程式 (系統和應用程式) 都存在輔助儲存器
  - 直到載入記憶體為止許多程式使用輔助儲存器作為它們處理時的來源和目的

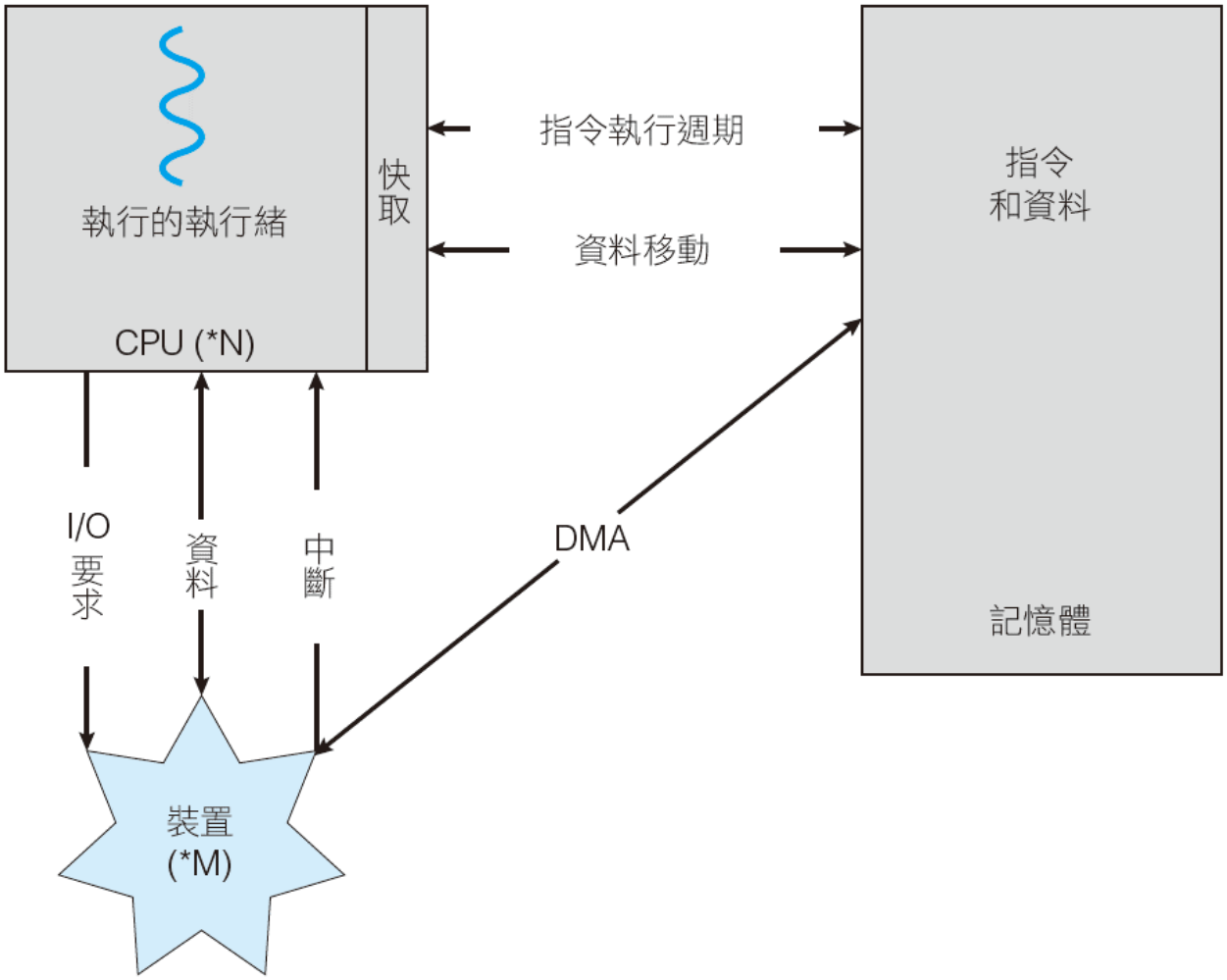
儲存裝置的階層


儲存容量
存取時間
更小
更大
更快
更慢
暫存器
快取記憶體
主記憶體
揮發性儲存
非揮發性儲存
非揮發性記憶體
硬碟機
光碟
磁帶
主儲存器
輔助儲存器
三級儲存器

I/O 結構

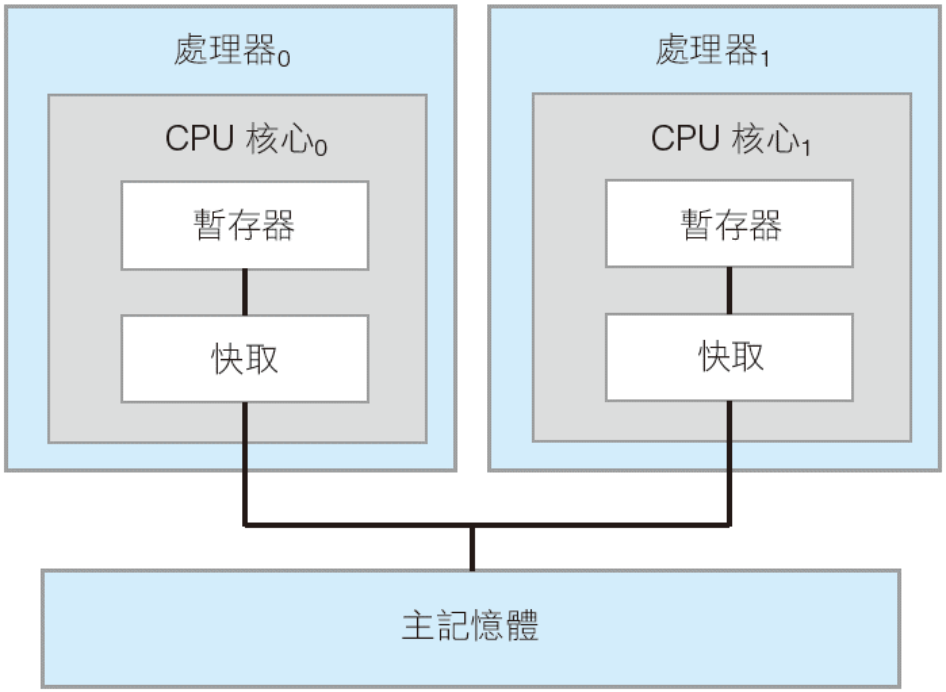
- 作業系統的大部份程式碼都用來管理 I/O，這是因為系統之可靠度和效能的重要性
- direct memory access 在設定 I/O 裝置的緩衝區、指標及計數器之後
  - 此裝置的控制器將一整個資料區塊從自己的緩衝儲存區直接傳送進或出的主記憶體，而不被 CPU 干涉
  - 每一區塊只產生一次中斷，而不像低速裝置，每個位元組均產生中斷一次
  - 當裝置控制器執行這些動作時，CPU 可以完成其它工作

現代電腦系統如何運作



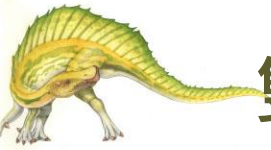
多處理器系統--對稱式多元處理架構

- 現代電腦從行動裝置到伺服器，其**多處理器系統** (multiprocessor system) 開始主宰運算領域

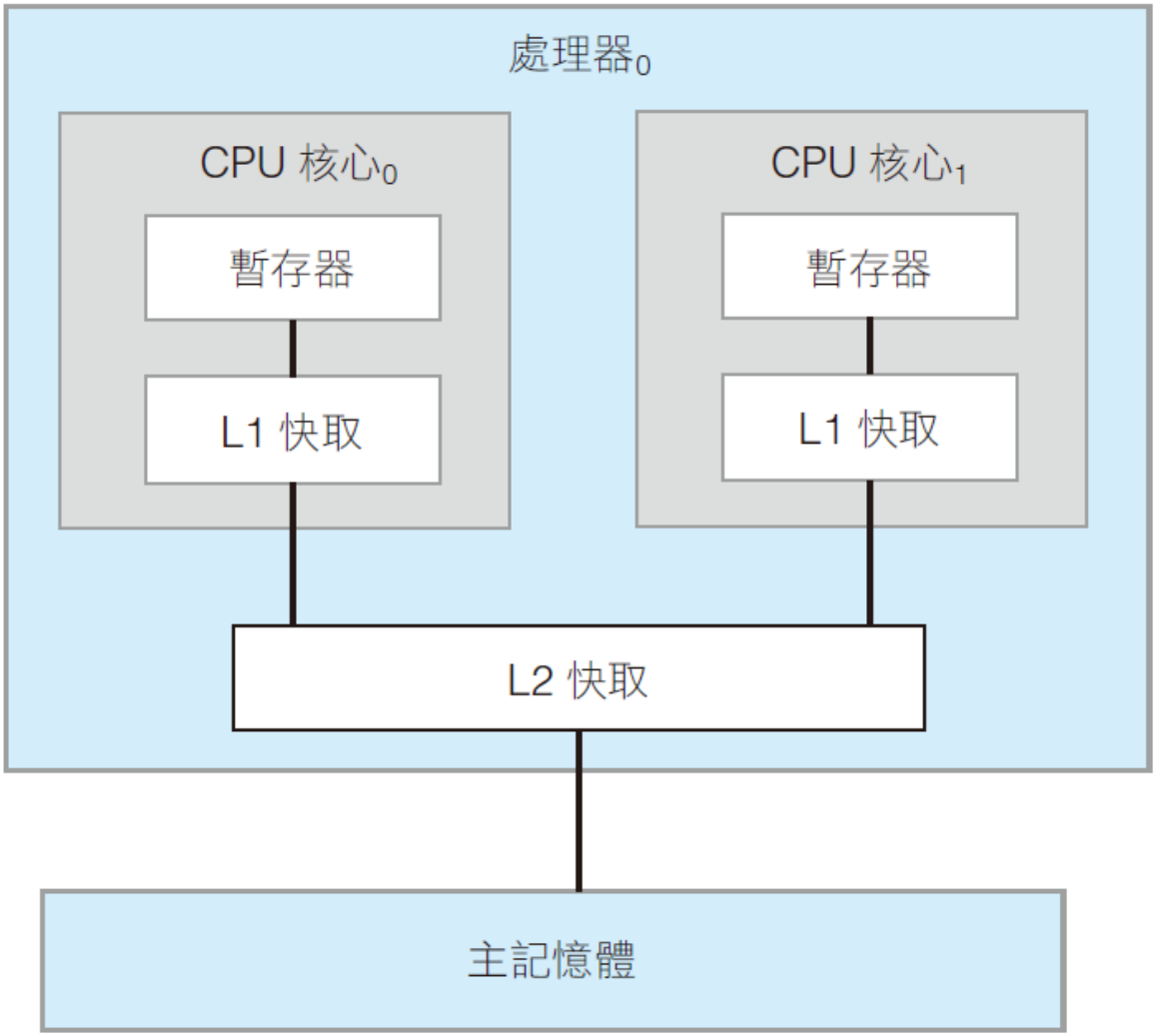


多處理器系統

- 多處理器的定義隨著時間的發展而有不同，現在的**多核心** (multicore) 系統就是在一個晶片中有多個運算核心
  - 多核心系統會比單核心系統的更有效率，因為在單一晶片上的通信會比晶片間通信更有效

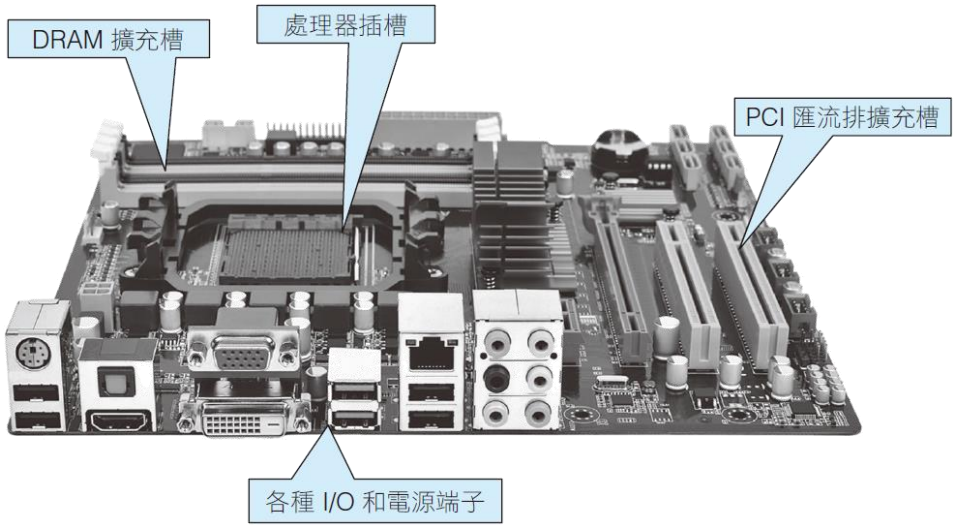


雙核心設計使用兩個核心在一個晶片內



叢集式系統

- PC 主板
  - 包含處理器插槽的桌上型電腦的主板
  - 當插槽插滿後，該板即為功能齊全的電腦。它由一個包含 CPU 的處理器插槽、DRAM擴充槽、PCIe 匯流排擴充槽和各種類型的 I/O 連接器組成。即使是低成本的一般 CPU 也都包含多個核心的處理器。有些主機板包含多個處理器插槽。更高級的電腦允許使用多個系統板來建立 NUMA 系統

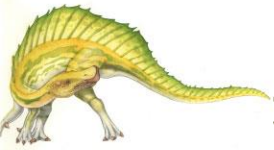


叢集式系統

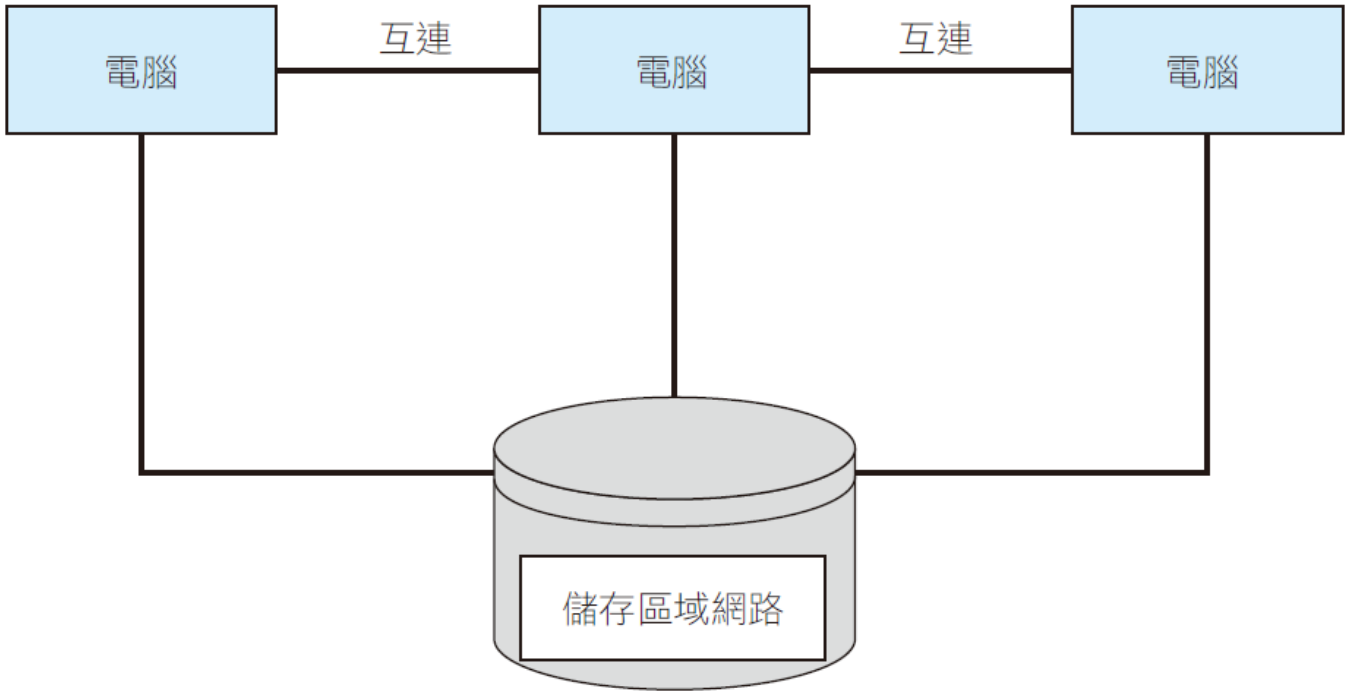
- **非對稱叢集系統** (asymmetric clustering) 中
  - 有一台機器處於**熱待機狀態** (hot-standby mode)
  - 另一台機器則正在執行應用程式
  - 熱待機的主機 (機器) 沒有做任何事情，它只是監督工作的伺服器
  - 如果伺服器壞了，熱待機主機就變成工作伺服器
- **對稱叢集系統** (symmetric clustering)
  - 兩台或更多台主機正在執行應用程式，並且它們互相監督

叢集式系統

- 這種型態的服務是普遍上所知道的**分散式上鎖管理者** (distributed lock manager, DLM)，包含在一些叢集技術中
- 這些改善可能藉由**儲存區域網路** (storage-area network, SAN)，SAN 允許許多系統連接到儲存單位



叢集式系統的一般結構



作業系統結構

- **多元程式** (multiprogramming) 規劃的目的就是讓 CPU 始終有工作做，以增加 CPU 的使用率
  - 並保持使用者滿意度，從而使 CPU 始終有一個要執行的程序
  - 在多元程式系統中，正在執行的程式稱為使用者**行程** (process)
- **多任務** (multitasking) 是一種多元程式的邏輯延伸
  - CPU 通過在多個行程之間切換來執行多個行程，但是切換快速發生而能提供使用者快速的**回應時間** (response time)
  - 在考慮到行程的執行時，它通常只執行很短的執行時間，直到它完成工作或需要執行 I/O 而結束
  - **虛擬記憶體** (virtual memory) 是達到這個目標的一個常用方法，這種技巧允許執行的工作可以不必全部放在記憶體中

多元程式系統的記憶體階層

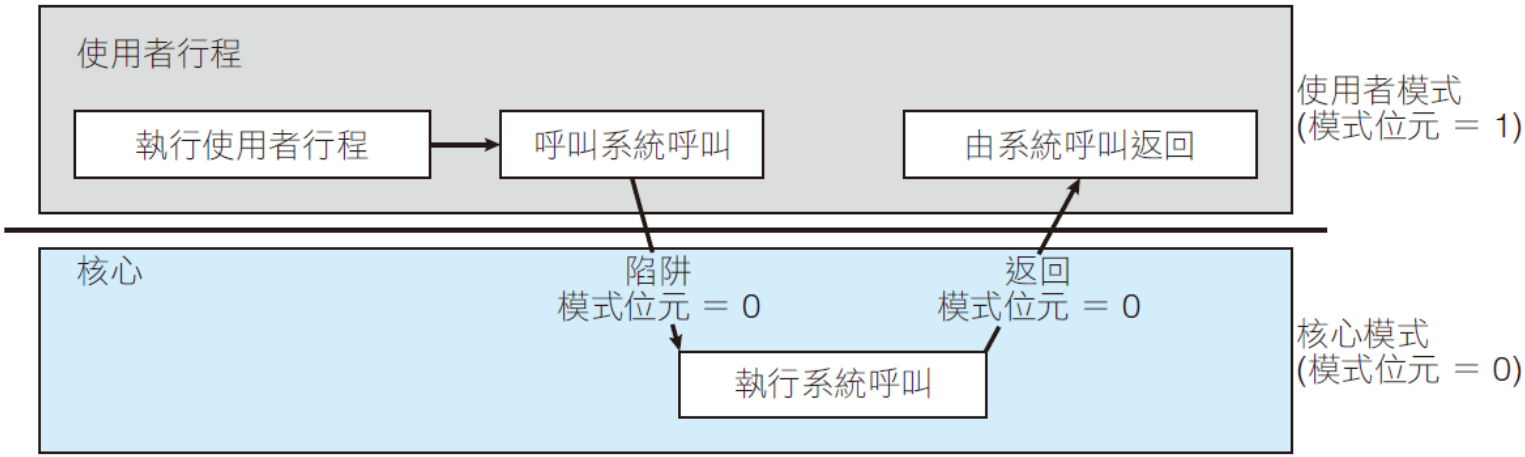
雙模式和多模式操作

- 兩種不同的運作模式 (mode)：**使用者模式** 和**核心模式**
  - 有一個稱為**模式位元** (mode bit) 的位元將加到電腦硬體之中，以便指出目前的模式：
    - 核心模式 (0)
    - 使用者模式 (1)
  - 有了模式位元，我們將可以區分某一個任務是在作業系統部份或是在使用者部份執行

雙模式和多模式操作

- 當電腦系統正在執行使用者應用程式部份，系統就在使用者模式
- 當使用者應用程式請求作業系統 (經由系統呼叫) 的服務時，則必須由使用者模式轉移到核心模式以實現這個需求
- 能夠支援虛擬化的 CPU 通常具有單獨的模式來指示，當虛擬機器管理程式 (virtual machine manager, VMM) 控制系統時，支援虛擬化的 CPU 通常有一個獨立的模式來指示

由使用者模式轉移到核心模式



雙模式和多模式操作

- 在系統啟動時，硬體由核心模式開始
  - 然後載入作業系統，接著在使用者模式開始執行使用者應用程式
  - 每當一個陷阱或中斷發生時，硬體從使用者模式轉換到核心模式 (也就是說，改變模式位元的狀態為 0)
  - 因此，每當作業系統得到電腦的控制權時，電腦就是在核心模式中
  - 使用者程式在將控制權交給一個使用者程式之前，總會轉換到使用者模式 (設定模式位元為 1)

雙模式和多模式操作

- 雙模式運作提供我們保護作業系統免於受到使用者之破壞—亦保護使用者免於受到相互間之干擾
  - 我們藉著指定一些可能引起上述危險的機器指令為特權指令 (privileged instruction)，以完成此項保護的工作
  - 硬體只允許這些特權指令於核心模式時執行
  - 如果企圖在使用者模式時執行特權指令，則硬體將不會執行這些指令，只將其看待成一個非法指令，而對作業系統發出中斷

資源管理

- 我們必須強調，程式本身並不是一個行程
  - 程式本身是一個被動的實體
    - 如同一個存在磁碟的檔案內容
  - 行程卻是一個主動實體
  - 單一執行緒的行程，擁有一個**程式計數器** (program counter) 指定下一個等待執行的指令

資源管理

- 在行程管理方面，作業系統必須負責下列的功能：
  - 使用者和系統行程的建立與刪除
  - 在 CPU 排班行程和執行緒
  - 行程的暫停和恢復
  - 提供行程同步的機制
  - 提供行程通信的機制

主記憶體管理

- 那些資料首先必須由 CPU 所產生的 I/O 呼叫傳送到主記憶體之中
  - 指令必須在主記憶體之中，CPU 才能執行它們
- 為了改善 CPU 使用率和電腦對使用者的回應速度
  - 一般用途電腦在記憶體上必須保持一些程式，這就產生記憶體管理需求

主記憶體管理

- 在記憶體管理方面，作業系統必須負責下列的功能：
  - 記錄正在使用的記憶體部份，以及是哪個行程在使用
  - 在需要時配置和回收記憶體空間
  - 決定哪些行程 (或部份的行程) 和資料要移入或移出記憶體空間
- 記憶體管理技術在第 9 章和第 10 章中有更詳盡的討論

檔案系統管理

- 為了電腦系統的方便使用，作業系統提供資訊儲存一個統一的邏輯觀點
- 作業系統轉移儲存裝置的實體特性定義成邏輯儲存單元，也就是檔案 (file)
- 作業系統映射檔案到實質媒體上，並且經由儲存裝置對這些檔案做存取

大量儲存器管理

- 由於主記憶體太小，以至於無法供應所有的資料和程式
  - 而且在電源消失時資料會隨之消失，電腦系統必須提供輔助儲存器以支援主記憶體
  - 作業系統必須負責下列的功能：
    - 安裝和卸載
    - 可用空間管理
    - 記憶體配置
    - 磁碟排班
    - 分割
    - 保護

大量儲存器管理

- 因為輔助儲存器經常且廣泛的使用，因此取決於輔助儲存器子系統和處理的演算法
- 同時，有許多使用的儲存器比輔助儲存器較慢，而且價格較低 (但有時有較大容量)
  - 磁碟資料備份、較少使用的資料和長期檔案儲存是其中一些例子
  - 磁帶機與磁帶、CD 與DVD、藍光機和光碟片是基本的三級儲存器 (tertiary storage) 裝置

快取管理

- **快取** (caching) 是電腦系統中的一個重要原理
  - 資訊通常存在某些儲存系統 (如主記憶體) 之中
  - 當我們需要一段特定的資訊時，應該先檢查該資訊是否在快取記憶體中
    - 如果是，資訊將直接由快取記憶體處取得
    - 否則，我們使用主儲存系統中的資訊，並將該資訊複製到快取記憶體之中

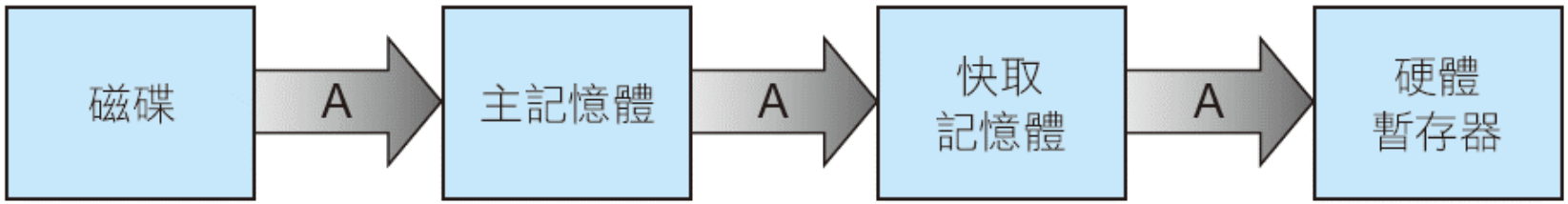
快取管理--不同階層儲存體的效能

- 由於快取記憶體的容量有所限制，**快取記憶體的管理** (cache management) 是一個設計上的重要問題
  - 快取記憶體大小和替代策略的細心選用，可以大幅提高效能

階層	1	2	3	4	5
名稱	暫存器	快取	主記憶體	固態硬碟	磁碟
基本大小	< 1 KB	< 16 MB	< 64 GB	< 1 TB	< 10 TB
製作技術	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
存取時間 (ns)	0.25 – 0.5	0.5 – 25	80 – 250	25,000 – 50,000	5,000,000
頻寬 (MB/sec)	20,000 – 100,000	5,000 – 10,000	1,000 – 5,000	500	20 – 150
被管理	編譯器	硬體	作業系統	作業系統	作業系統
被備份	快取	主記憶體	硬碟	硬碟	硬碟或磁帶

快取管理--整數 A 從磁碟到暫存器的轉移過程

- 這個問題叫作**快取記憶體的一致性** (cache coherency)，而且通常是硬體的問題 (由作業系統以下的層次處理)
- 因為不同的副本可以同時地存取和更新，我們必須保證當某一地的副本被更新時
  - 所有其它副本也要盡快地更新

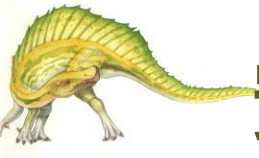


I/O 系統管理

- 作業系統的目的之一是對使用者隱藏特定硬體裝置的性質
  - 在 UNIX 中，I/O 裝置的特點由作業系統的 I/O **子系統** (I/O subsystem) 隱藏起來
  - I/O 子系統包含有幾個元件：
    - 包括緩衝、快取和連線，以及周邊作業的記憶體管理元件
    - 通用裝置驅動程式介面
    - 特定硬體裝置驅動程式

安全與保護

- **保護** (protection) 是一種控制行程或使用者存取電腦系統定義之資源的機制
  - 這個機制必須提供此控制被加入和實行此控制的方法
  - 保護可以藉由偵測子系統元件間介面的潛在錯誤來改善可靠度
  - 介面錯誤的及早偵測通常能夠避免正常子系統受到功能受損子系統的影響
- **安全** (security) 的工作是防護系統免於外部與內部的攻擊



安全與保護

- 保護和安全需要系統能夠區別它的所有使用者
  - 大部份作業系統維護一個使用者名字和相關的**使用者識別碼** (user identifier, user ID) 的串列
  - 以 Windows 的說法，這就是**安全性識別碼** (security ID, SID)
- 群組功能可以被製作成一整個系統的群組名字和**群組識別碼** (group identifier) 串列
- 有時候一個使用者需要**提升權限** (escalate privilege) 以獲得某一行為的額外許可

虛擬化

- 虛擬化允許作業系統在其它作業系統內以應用程式方式執行
- **模擬** (emulation) 被用在當原始 CPU 型態和目的 CPU 型態不相同時
  - 如當 Apple 將它的桌上型和筆記型電腦從 IBM Power CPU 轉換到 Intel x86 CPU 時，它包含一個模擬機制
- 使用虛擬化 (virtualization)，原本為某一特殊 CPU 架構編譯的作業系統在另一作業系統內執行也原生於此CPU
- VMware 應用程式是**虛擬機管理器** (virtual machine manager, VMM)
  - VMM 運行客戶作業系統，管理其資源使用並保護每個來賓免受其它來賓侵害

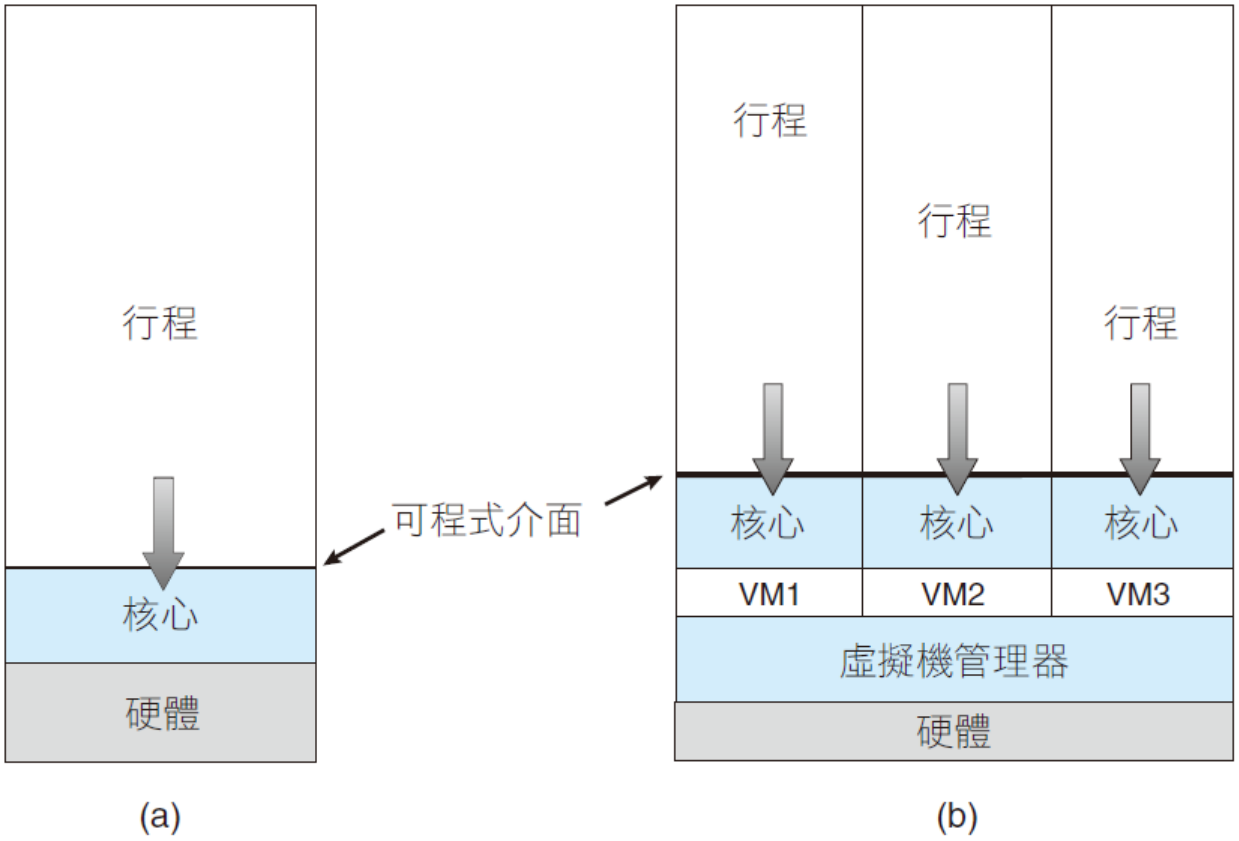
虛擬化

- 即使現代的作業系統可以完全可靠地執行許多應用程式，虛擬化的使用依然在成長
  - 在筆記型電腦和桌上型電腦上，一個 VMM 允許使用者安裝許多作業系統作為探討或執行不同於原始主機之作業系統所寫的應用程式
  - 例如，一台 x86 CPU 執行 macOS 的Apple 筆記型電腦可以執行一個 Windows 10 guest 以便執行 Windows 的應用程式
  - 為多個不同作業系統撰寫軟體的公司可以在單一台實體伺服器執行所有這些作業系統，以便發展、測試和偵錯
  - 在資料中心，虛擬化變成一種執行和管理運算環境的通用方法

虛擬化

- 類似於 VMware ESX 和 Citrix XenServer 不再執行於主機作業系統上，而是在主機作業系統執行，為虛擬機行程提供服務和資源管理

電腦運行



(a) 單一作業系統　　(b) 三個虛擬機

分散系統

- 分散系統是一組實體上分開的電腦系統 (可能是不同的電腦系統)，以網路連結的集合，以提供使用者存取系統所維護的各種不同資源
- **TCP/IP** 是最普遍的網路協定
- **區域網路** (local-area network, LAN) 連接在一個房間、一棟建築物或一個校園內的電腦
- **廣域網路** (wide-area network, WAN) 則通常連接建築物、城市或國家
  - 例如，全球化的公司可能有 WAN 連接它在全世界的辦公室

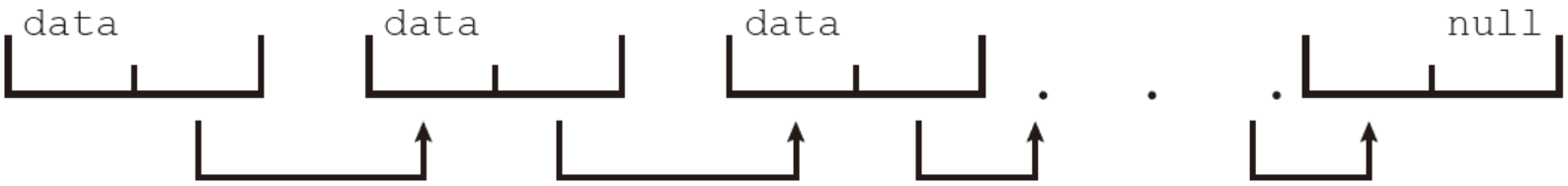
分散系統

- 這些網路可能執行一種協定或數種協定。新技術不斷的來臨帶來新的網路形式
  - **都會網路** (metropolitan-area network, MAN) 可以連結城市內的建築物

- 網路作業系統 (network operating system) 是一種能提供以下特性的作業系統
  - 例如跨越網路間的檔案分享，以及允許不同電腦上不同行程間交換訊息的通信技巧

串列、堆疊和佇列

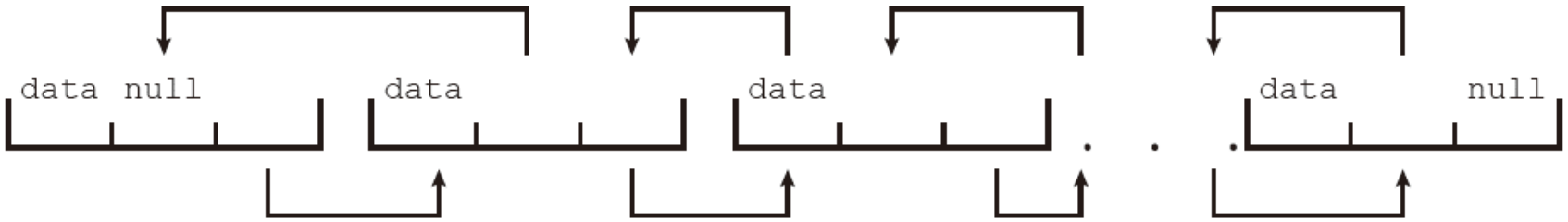
- 在單向鏈結串列 (singly linked list)，每一項指向它的下一項。



單向鏈結串列

串列、堆疊和佇列

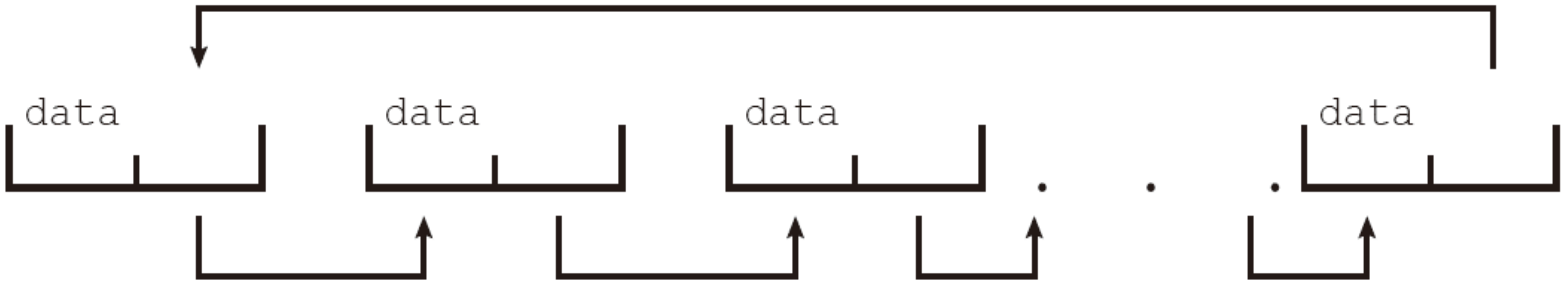
- 在雙向鏈結串列 (doubly linked list)，每一項指向它的前一項和下一項。



雙向鏈結串列

串列、堆疊和佇列

- 在環狀鏈結串列 (circularly linked list)，串列的最後一項指向串列的第一項，而不是指到空項目 (null)。

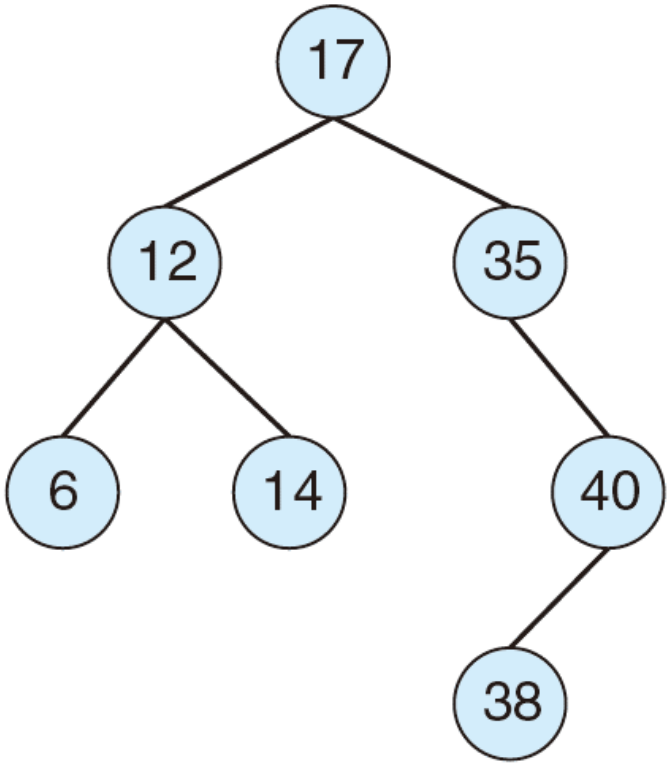


環狀鏈結串列

樹

- Tree
  - 父－子關係鏈結
  - 在**一般的樹** (general tree)，父節點可以有無限個子節點
  - 在**二元樹** (binary tree)，父節點至多有 2 個子節點，我們命名為*左子節點* (left child) 和*右子節* (right child)
  - **二元搜尋樹** (binary search tree) 額外要求在父節點的 2 個子節點間順序，*left_child* <= *right_child*

二元搜尋樹



位元映像

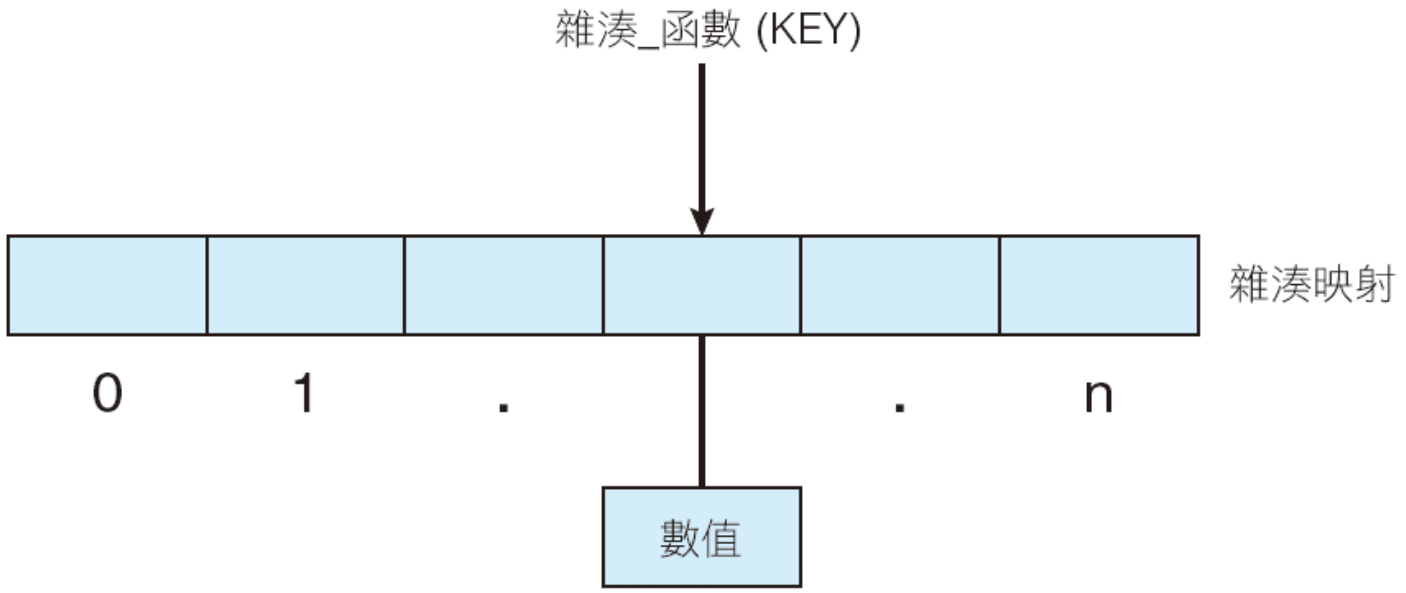
- **位元映像** (bitmap) 是可以用來表示 *n* 個項目狀態的一串 *n* 位元二進位數字
  - 假設我們有一些資源，每一資源的可利用性是以二位元數表示：
    - 0 表示資源可取得，而 1 表示資源是不可取得 (或是相反)
  - 位元映像的第 *i* 個位置數值關聯到第 *i* 個資源
    - 考慮以下的位元映像

0 0 1 0 1 1 1 0 1

位元映像

- Linux 核心資料結構
  - 在 Linux 核心所使用的資料結構可以在核心原始碼獲得
  - 包含檔案 <linux/list.h> 提供在整個核心使用鏈結串列資料結構。在 Linux 被稱為 kfifo 的佇列，和它的實作可以在原始碼 kernel 目錄的檔案 kfifo.c 找到
  - Linux 也提供一個使用紅黑樹 (red-black tree) 的平衡二元搜尋樹製作
  - 細節可以在包含檔 <linux/rbtree.h> 找到

雜湊映射



傳統運算

- **入口網站** (portal)，它提供網頁存取到它們內部伺服器
  - 網路電腦 (network computer)，或精間客戶 (thin client)
    - 主要是能瞭解網頁為基礎之運算的終端機
    - 在需要更多安全和容易維護時，它們被用來取代的傳統的工作站
  - 行動電腦可以和 PC 同步以允許公司資訊易於攜帶使用
  - 它們也可以連接到**無線網路** (wireless network) 以使用公司的入口網站(以及其它網頁資源)

傳統運算

- 許多家庭使用**防火牆** (firewall) 以保護他們的網路免於安全漏洞
  - 防火牆限制了設備和網路之間的通信

行動運算

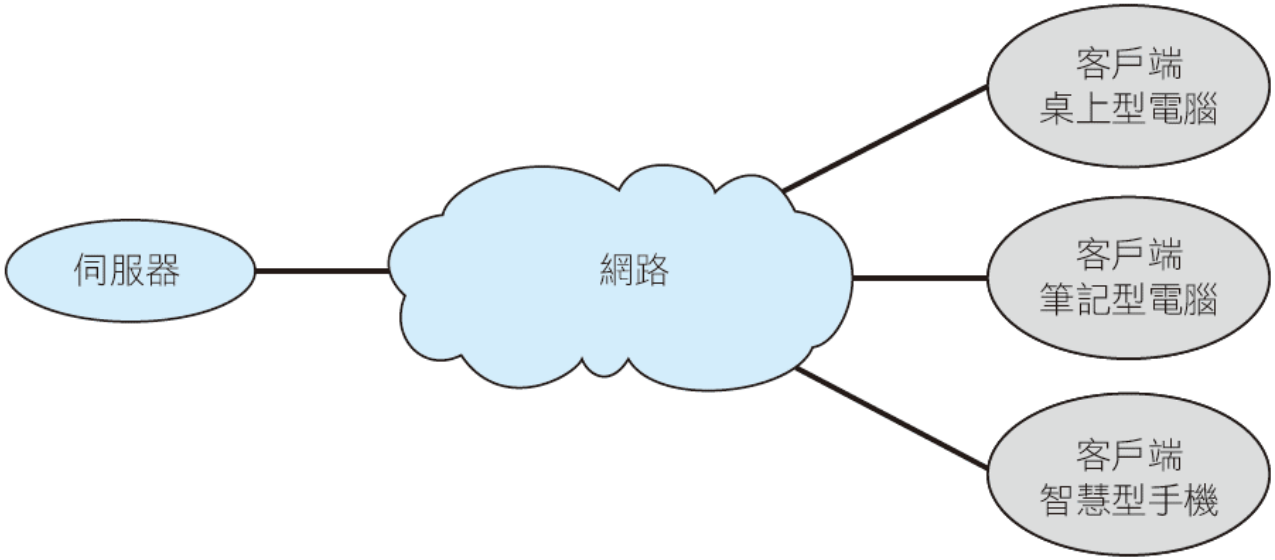
- **行動運算** (mobile computing) 是指在手持式智慧型手機和平板電腦上的運算
  - 這些裝置有可攜性和輕量化的獨特物理特性
  - 行動系統與桌上型和筆記型電腦相比
    - 行動系統放棄了螢幕大小、記憶體容量和整體的功能性
    - 以取得手持行動存取電子郵件與網頁瀏覽等服務
      - 全球定位系統 (global positioning system, GPS) 晶片、加速度器和陀螺儀

行動運算

- 也許這些特性的另一個實際使用是在擴增實境的應用
  - 擴增實境是將資訊疊加在目前環境的顯示上
- 行動裝置通常使用 IEEE 標準 802.11 無線或是蜂巢式資訊網路
- 目前主宰行動運算的兩個作業系統是：
  - **Apple iOS**
  - **Google Android**
- iOS 是設計在Apple iPhone 和 iPad 行動裝置上執行

客戶端–伺服器運算

- 現在的網路結構的特徵在於**伺服器系統** (server systems) 能夠滿足**客戶端系統** (client systems) 的工作要求
- 這種形式的專用分散式系統稱為**客戶端–伺服器** (client-server) 系統具有一般的架構



客戶端–伺服器系統的一般架構

客戶端–伺服器運算

- **運算伺服器系統** (compute-server system) 提供介面給客戶端
  - 客戶端可以送出要求以執行一項動作
    - 如讀資料
  - 伺服器對此動作的反應是執行動作，並且送回結果給客戶端
  - 回應給客戶端資料要求的執行資料庫伺服器就是這種系統的一個範例

客戶端–伺服器運算

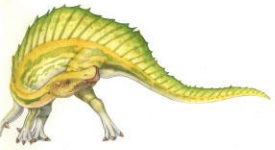
- **檔案伺服器系統** (file-server system) 提供介面給客戶
  - 客戶可產生、更新、讀取和刪除檔案
  - 這種系統的一個範例就是執行網頁的瀏覽器

點對點運算

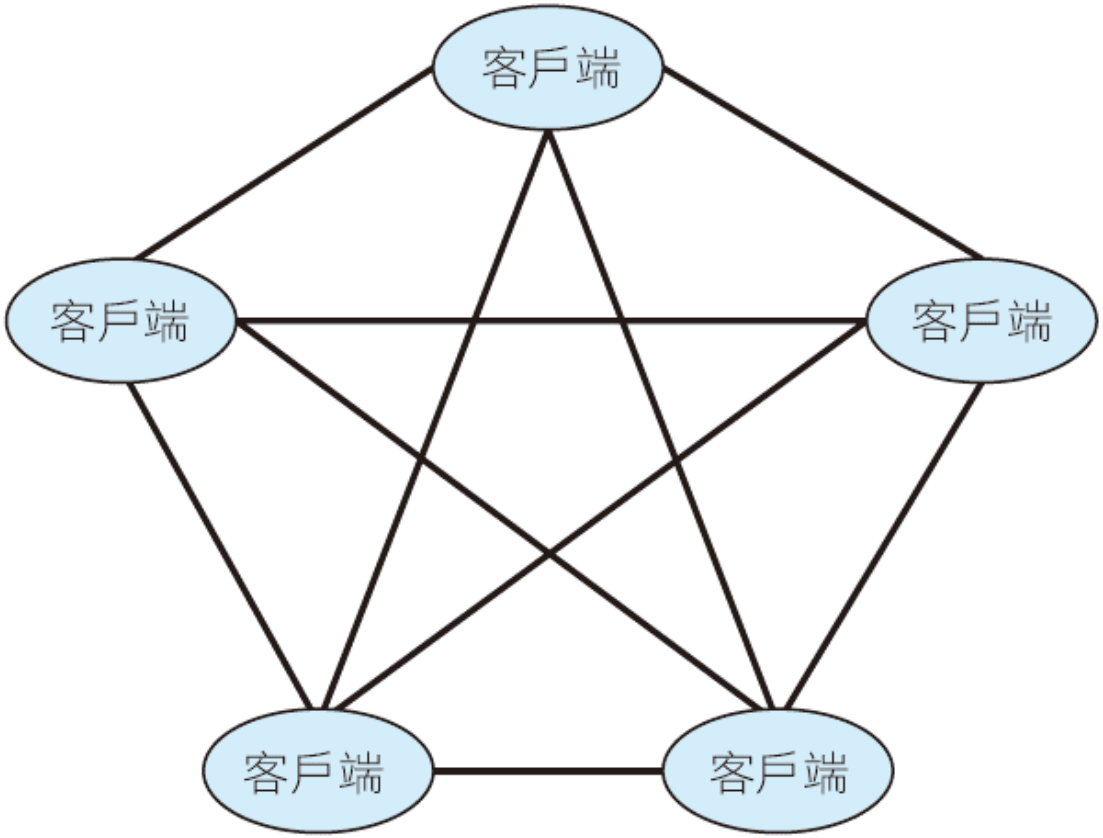
- 分散式系統的另一種結構是點對點 (peer-to-peer, P2P) 系統模式
- 點對點系統，節點必須先加入點對點網路
  - 一旦節點加入網路後，它可以開始提供服務給網路和由網路其它節點要求服務
  - 判斷有哪些服務可用，可以從兩種通用方法中的一種來完成：
    - 當節點參與網路，以集中查詢服務在網路註冊它的服務。任何節點需要特定的服務時
      - 首先接觸這個集中查詢服務，來決定哪一個節點提供此服務
      - 剩下來的通信發生在客戶端和服務提供者之間

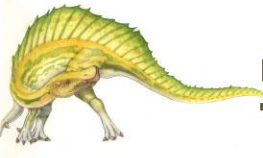
點對點運算

- 另一種替代方法是使用非集中查詢服務
  - 作為客戶端的節點必須藉由廣播服務要求給網路上所有其它的節點，以發現哪一個節點可提供所需要的服務
  - 可提供該服務的節點 (或節點群) 回應提出需求的節點
  - 為了支持這個方法，必須提供搜尋協定(discovery protocol)，讓節點群發現在網路上其它點所提供的服務
- Skype 允許客戶端使用一種被稱為 VoIP (voice over IP) 的技術，經由網際網路撥打語音電話、視訊電話和傳送簡訊



沒有集中式服務的點對點系統





雲端運算

- **雲端運算** (cloud computing) 是一種經由網路提供計算、儲存，甚至是應用作為服務的一種運算
- 在某些形式上，它是虛擬化的邏輯延伸，因為它使用虛擬化作為其功能的基礎
  - 例如，**亞馬遜彈性運算雲** (Amazon Elastic Compute Cloud, ec2) 的設備有數以千計的伺服器、數百萬個虛擬機和千兆位元組 (petabytes) 的儲存空間，給任何人在網際網路上使用
  - 使用者每月付的費用是根據他們使用了多少這些資源作為依據

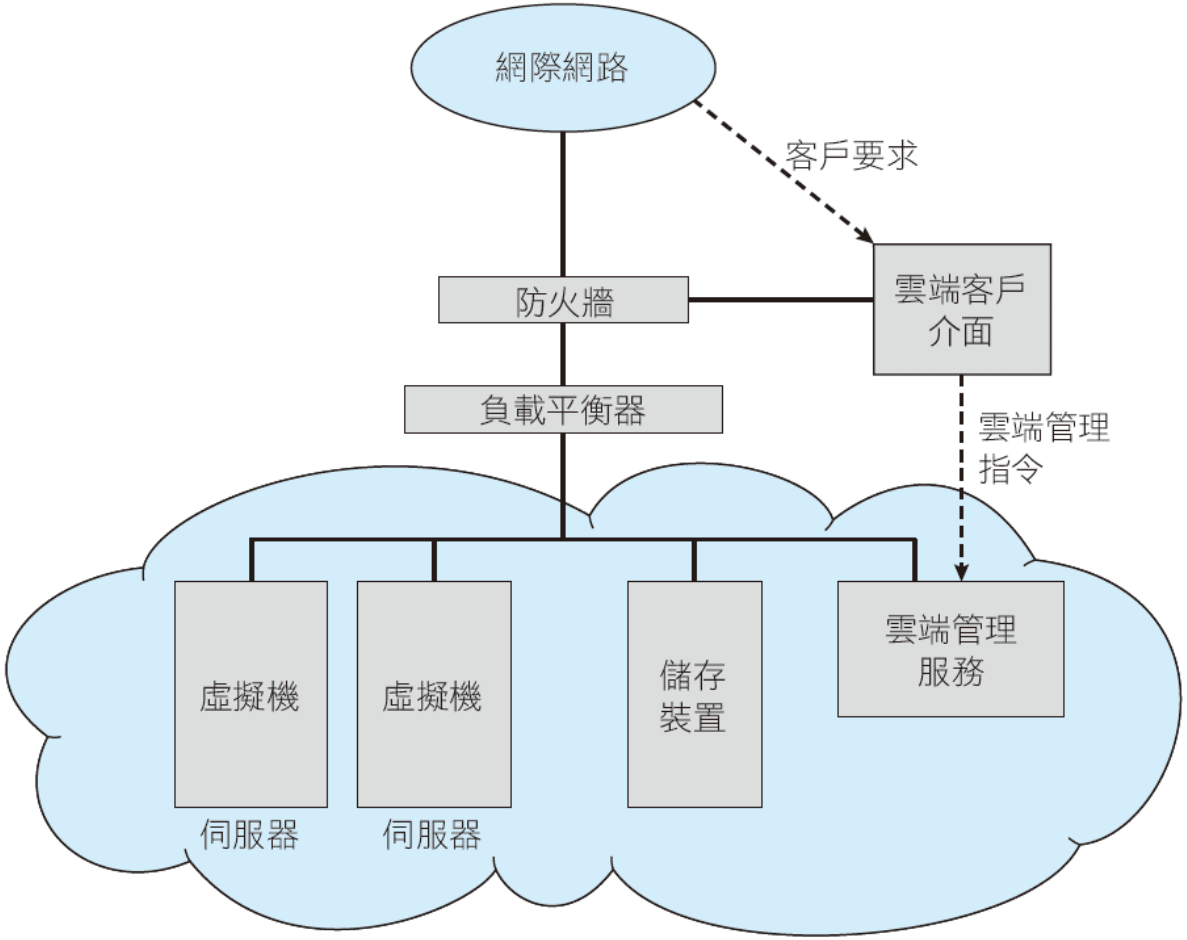
雲端運算

- 許多型態的雲端運算：
  - **公用雲** (public cloud)
    - 經由網際網路給任何願意為此服務付費者取得的雲
  - **私有雲 (private cloud)**
    - 由公司自行經營並自己使用的雲
  - **混合雲 (hybrid cloud)**
    - 包含公用雲和私有雲混合的雲
  - **軟體即服務 (software as a service, SaaS)**
    - 經由網際網路取得文書處理程式或試算表

雲端運算

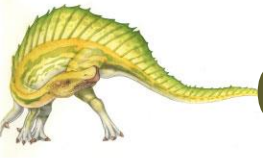
- **平台即服務** (platform as a service, PaaS)
  - 經由網際網路準備堆疊
    - » 例如資料庫伺服器
- **基礎設施即服務 (infrastructure as a service, IaaS)**
  - 在網際網路可取得的伺服器或儲存器
    - » 例如，可以作為生產資料製作備份的儲存器

雲端運算


網際網路
客戶要求
防火牆
雲端客戶介面
負載平衡器
雲端管理指令
虛擬機
虛擬機
儲存裝置
雲端管理服務
伺服器
伺服器

即時嵌入系統

- 嵌入式系統幾乎都是執行**即時作業系統** (real-time operating system)
  - 即時系統是使用在當對於處理器的操作或資料的傳送在時間要求上很嚴謹時
    - 因此，它通常是用在特定應用的控制裝置
  - 感應器將資料傳送給電腦
  - 電腦必須將資料加以分析
- 即時作業系統有著定義嚴謹的固定時間限制
  - 行程必須在所定義的時間限制內完成，否則系統將失效



GNU

- 1985 年，斯托曼發表了 GNU 宣言，該宣言認為所有軟體都應該是免費的
  - 他還成立了自由軟體基金會 (Free Software Foundation, FSF)，旨在鼓勵使用和開發自由軟體
- 考慮 GNU/Linux 開放原始碼作業系統的範例

作業系統研究

- 虛擬化作為一種主流 (通常是免費的) 的電腦功能的興起，使得在一個核心系統之上運行許多作業系統的方式變成為可能
  - VMware (http://www.vmware.com) 為Windows 提供了一個免費的“播放器”，可以在其上運行數百個免費的“虛擬設備”
- Virtualbox (http://www.virtualbox.com) 在許多作業系統上提供了免費的開源虛擬機管理器
- 使用此類工具，學生可以在沒有專用硬體的情況下試用數百種作業系統

END!