Chapter 1：概説
Introduction

Chapter 1: Introduction

- 作業系統做些什麼事(What Operating Systems Do)
- 電腦系統組織(Computer-System Organization)
- 電腦系統架構(Computer-System Architecture)
- 作業系統結構(Operating-System Structure)
- 作業系統的操作(Operating-System Operations)
- 行程管理(Process Management)
- 主記憶體管理(Memory Management)
- 儲存體管理(Storage Management)
- 保護與安全(Protection and Security)
- 核心資料結構(Kernel Data Structures)
- 運算環境(Computing Environments)
- 開放原始碼作業系統(Open-Source Operating Systems)

章節目標

- 描述電腦系統的基本組織
- 提供作業系統主要單元的導覽
- 介紹多種類型的電腦運算環境的概觀
- 探討一些開放原始碼的作業系統

章節重點

- 作業系統扮演的角色及主要工作
- 電腦系統組織及硬體運作情形
  - Polling, 中斷機制
  - Memory hierarchy
  - I/O operating: Direct memory access
- 多元程式規劃及分時系統概念
- 作業系統運作模式
  - 雙模式運作
  - 系統呼叫
- Cache and caching
- 電腦運作環境
  - 虛擬機，雲端計算，即時系統
- 開放原始碼OS

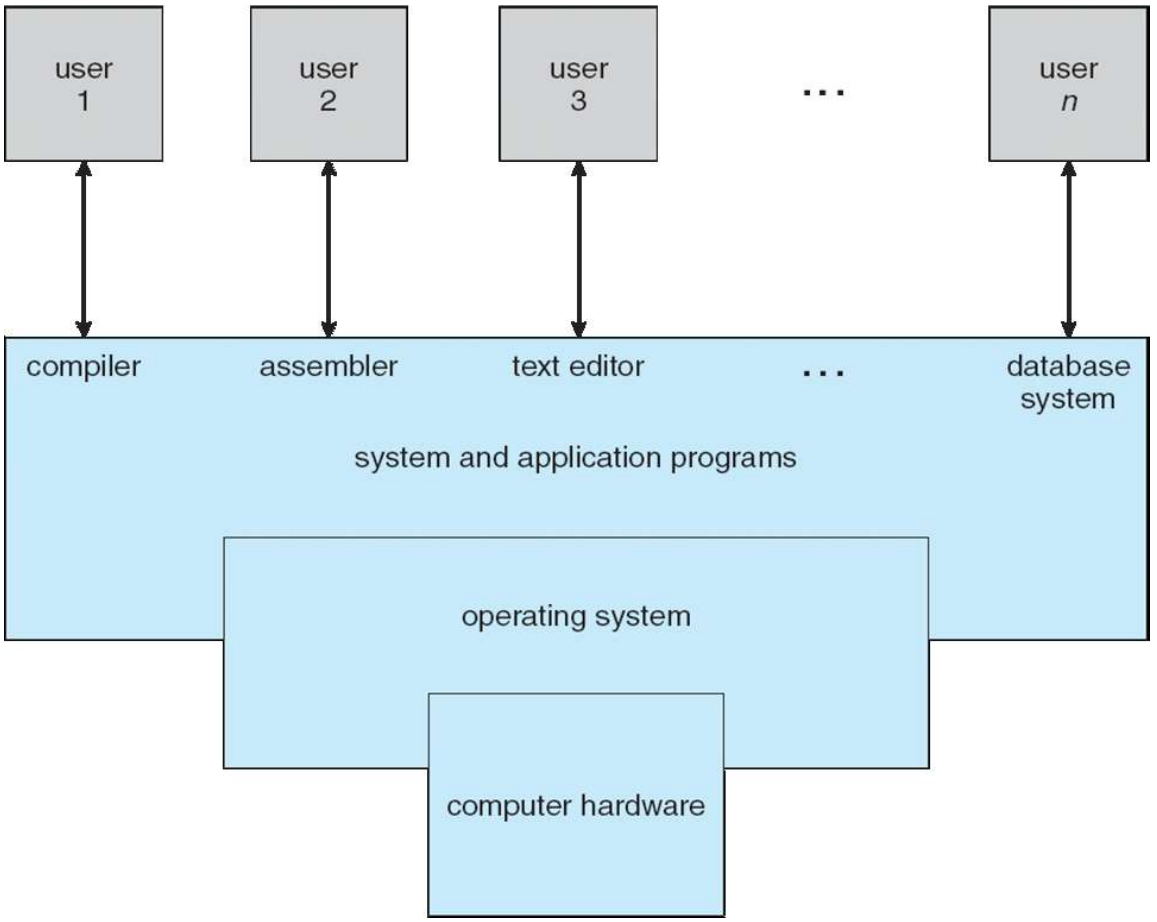
What is an Operating System?

- 作業系統扮演電腦使用者和電腦硬體間的中介程式，管理電腦之硬體資源讓程式有效率執行的軟體，它也提供使用者方便使用的環境。
- 作業系統是電腦系統的核心系統。
- 作業系統的目的:
  - 執行使用者程式，並讓使用者更容易解決問題(easier)
  - 讓電腦系統方便使用(convenient)
  - 以有效率(efficient)的方式使用電腦硬體

作業系統做些什麼事？

- 電腦系統可分成四大組成單元
  - 硬體(hardware) – 提供基本運算資源，如：CPU, 記憶體, I/O 裝置等
  - 作業系統(OS)
    - 控制和協調應用程式和使用者對於硬體的使用
  - 應用程式(Application programs) – 決定系統資源的使用方式以解決使用者愈處理的問題
    - 文書處理器, 編譯器, 網頁瀏覽器, 資料庫系統, 視訊遊戲
  - 使用者(Users)
    - 人, 機器, 其他電腦

電腦系統四大組成單元間的關係



作業系統做些什麼事？(續)

- 可分別從使用者觀點(User View)及系統觀點(System View)來了解OS所扮演的角色
- 使用者觀點(User View)
  - PC使用者：主要希望方便及容易使用為主，會希望更好的效能，但較不關心資源的使用率(Resources Utilization) 。
  - 大型主機(mainframe)或迷你電腦使用者：由於同時會有許多使用者會透過終端機來使用，希望將資源使用率達到極大化。
  - 工作站(workstation)使用者：使用便利性(Usability) 與資源使用率 (Resources Utilization) 之間的妥協安排。
  - 行動電腦(mobile computer)使用者：除使用便利性外，需特別考慮電池的耗能問題
  - 嵌入式電腦(embedded computers)使用者：不須使用者介面

作業系統做些什麼事？(續)

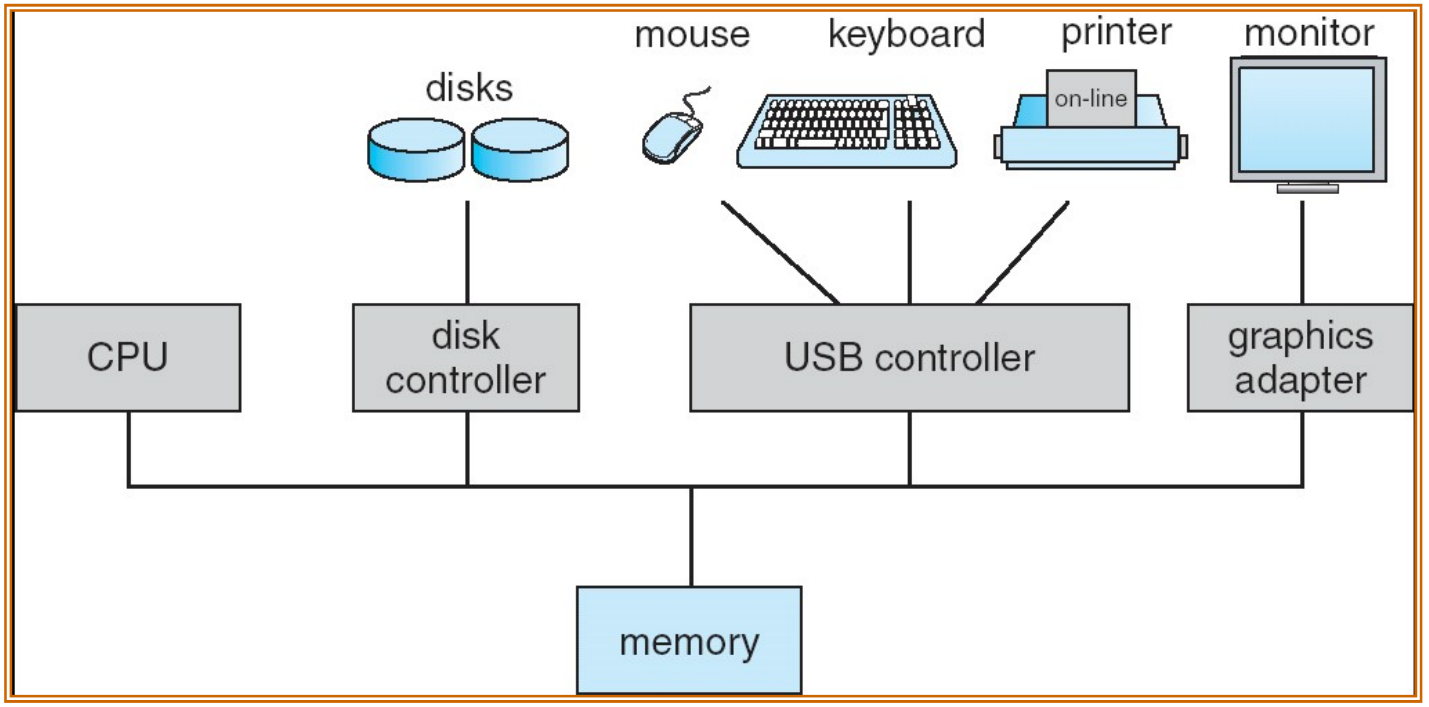
- 系統觀點(System View)
  - 作業系統是一個資源分配者(resource allocator)
    - 管理所有的資源
    - 當資源的使用有衝突時，決定如何有效率和公平的進行資源使用
  - 作業系統是一個控制程式(control program)
    - 控制程式的執行以避免錯誤和電腦的不當使用

作業系統的定義

- 沒有普遍接受的定義
- 一般說法：“當您訂購一套作業系統時，供應商交貨包含的所有東西”均可視為作業系統，但差異可能很大。
- 核心(kernel):作業系統最主要的部份，電腦開機後即常駐於記憶體中，負責最關鍵的功能，包括行程管理、記憶體管理、I/O管理及行程間的通訊等。
- 其餘的程式為系統程式(和作業系統一起販售)或應用程式。

電腦系統組織(Computer System Organization)

- 一個或多個CPU及裝置控制器(device controllers)經由共用匯流排(bus)存取主記憶體，因併行式 (Concurrently) 執行的關係，會因共用記憶體而競爭記憶體時脈週期 (Memory Cycle)。

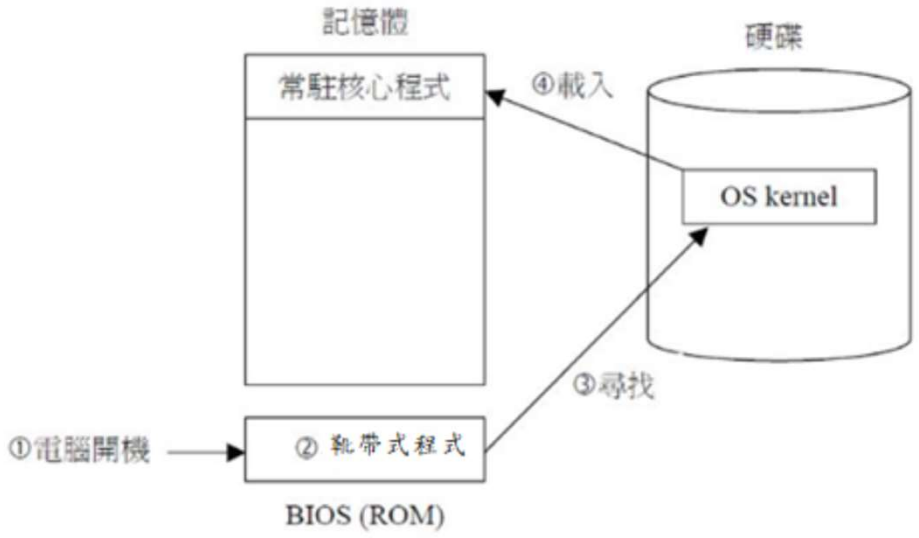


電腦系統的運作情形

- I/O 裝置和CPU可以並行地執行
- 每一個裝置控制器負責一種特殊裝置型態
- 每一個裝置控制器有一個區域緩衝區(local buffer)
- CPU 在主記憶體和區域緩衝區間進行資料移動
- I/O是在外部裝置與控制器的區域緩衝區間進行
- 當I/O的動作完成時，裝置控制器藉由中斷(interrupt)通知CPU。
- 中斷(interrupt) 可經由硬體及軟體來觸發

電腦啟動

- 靴帶式程式(bootstrap program)
  - 在電源開啟或重新啟動時第一個被載入執行的程式
  - 通常儲存在唯讀記憶體(read-only memory,ROM)或電子抹除式可複寫唯讀記憶體（Electrically Erasable Programmable Read Only Memory，EEPROM）中，統稱為韌體(firmware)。
  - 主要工作為初始化系統(包括CPU暫存器、裝置控制器及主記憶體內容等)及載入作業系統核心並開始執行。



Boot sequence of IBM-PC compatibles

1. Runs the instruction located at the physical memory address FFFFFFF0h of the BIOS (a jump instruction to the BIOS start-up program
2. POST (Power On Self Test) to check and initialize required devices such as DRAM and the PCI bus
3. Goes through a pre-configured list of non-volatile storage devices ("boot device sequence") until it finds one that is bootable. A bootable device is defined as one that can be read from, and where the last two bytes of the first sector contain the little-endian word AA55h (also known as the MBR(Master Boot Record) boot signature).
4. Loads the boot sector to linear address 7C00h and transfers execution to the boot code. (this boot code is not operating-system specific)
5. The conventional MBR code checks the MBR's partition table for a partition set as bootable. If an active partition is found, the MBR code loads the boot sector code from that partition, known as Volume Boot Record (VBR), and executes it.
6. The VBR is often operating-system specific; however, in most operating systems its main function is to load and execute the operating system kernel, which continues startup.

中斷的類型

- 硬體中斷
  - I/O 裝置要求傳送資料
  - I/O 裝置已完成資料傳送
  - 計時器時間到了
  - …
- 軟體中斷
  - 陷阱(trap)或例外(exception)：由於非法的使用或錯誤的使用指令或資料
    - 暫存器溢位 (Register Overflow)
    - 堆疊溢位 (Stack Overflow)
    - 使用不適當的操作指令 (Invalid Operation Code)
    - 運算式中有除以零 (Divided By Zero) 的情形
    - 違反系統之使用限制 (Protection Violation)
  - 由指令所引發的中斷。如：系統呼叫 (Supervisor Call 或稱 System Call)

中斷的處理機制

- 中斷的處理會經由中斷向量(interrupt vector)轉移控制到中斷服務常式(interrupt service routine, ISR)
- 每一種型態的中斷都有相對應的中斷服務常式，決定應該採取什麼動作，中斷向量是存放著中斷服務常式的位址的表格 。
- 會利用堆疊 將被中斷之指令的位址和執行狀態儲存起來，在中斷處理結束後再繼續執行。
- 現有作業系統是採取中斷驅動(interrupt-driven)方式實現出來

中斷 (Interrupt) 發生時的處理流程

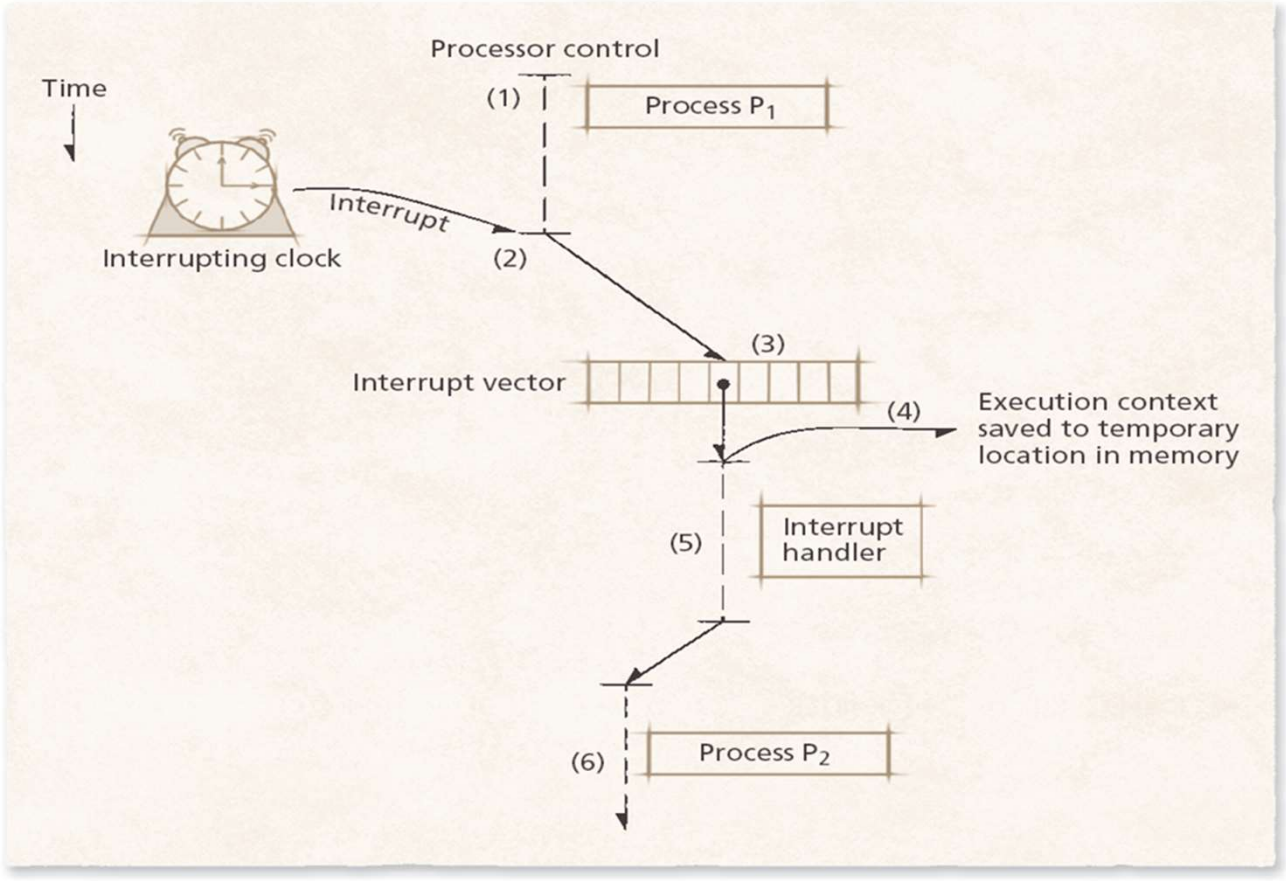
- 中斷 (Interrupt) 發生時的處理流程：
  - 系統(CPU)暫停目前行程(process)的執行，同時保存其當時執行的狀態，包括CPU暫存器及程式計數器(program counter) 等。
  - 根據中斷編號去查詢中斷向量，以便可以找到相對應中斷服務常式之起始位址。
  - 跳到指定的中斷服務常式起始位址，開始執行。
  - 中斷服務常式執行完畢後，將控制權交回給OS。
  - OS選擇接下來要執行的其他行程，或是回到被中斷的行程繼續執行。

x86 Interrupt Vectors

- provides 256 interrupts
  - 15 Hardware IRQ's plus one Non-Maskable IRQ
  - Others for software interrupts and exception handlers
- situated at 0000:0000 which extends for 1024 bytes

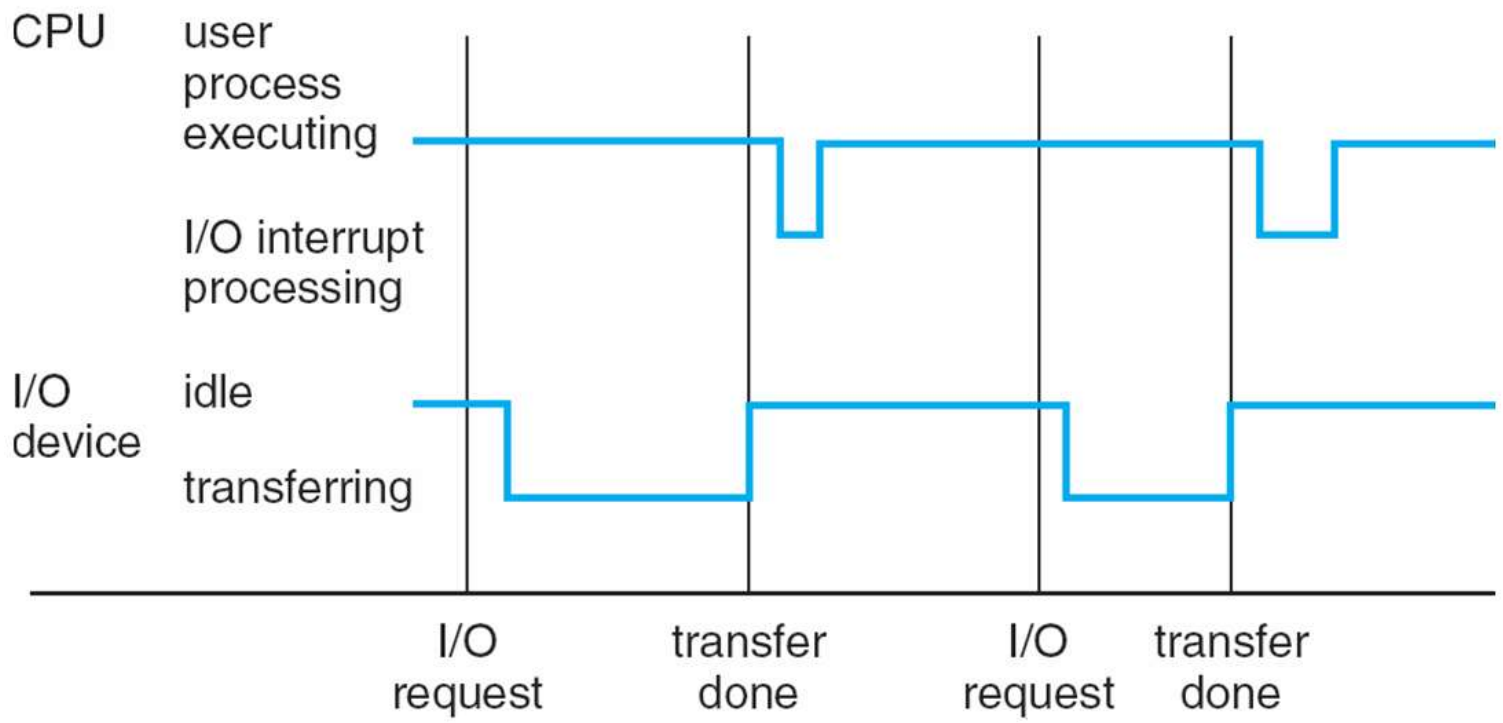
INT (Hex)	IRQ	Common Uses
00 - 01	Exception Handlers	
02	Non-Maskable IRQ	Non-Maskable IRQ (Parity Errors)
03 - 07	Exception Handlers	
08	Hardware IRQ0	System Timer
09	Hardware IRQ1	Keyboard
0A	Hardware IRQ2	Redirected
0B	Hardware IRQ3	Serial Comms. COM2/COM4
0C	Hardware IRQ4	Serial Comms. COM1/COM3
0D	Hardware IRQ5	Reserved/Sound Card
0E	Hardware IRQ6	Floppy Disk Controller
0F	Hardware IRQ7	Parallel Comms.
10 - 6F	Software Interrupts	
70	Hardware IRQ8	Real Time Clock
71	Hardware IRQ9	Redirected IRQ2
72	Hardware IRQ10	Reserved
73	Hardware IRQ11	Reserved
74	Hardware IRQ12	PS/2 Mouse
75	Hardware IRQ13	Math's Co-Processor
76	Hardware IRQ14	Hard Disk Drive
77	Hardware IRQ15	Reserved
78 - FF	Software Interrupts	

Interrupt處理流程示意圖



- Source: H.M. Deitel, Operating Systems, 3rd Ed.

單一行程Interrupt 時序圖



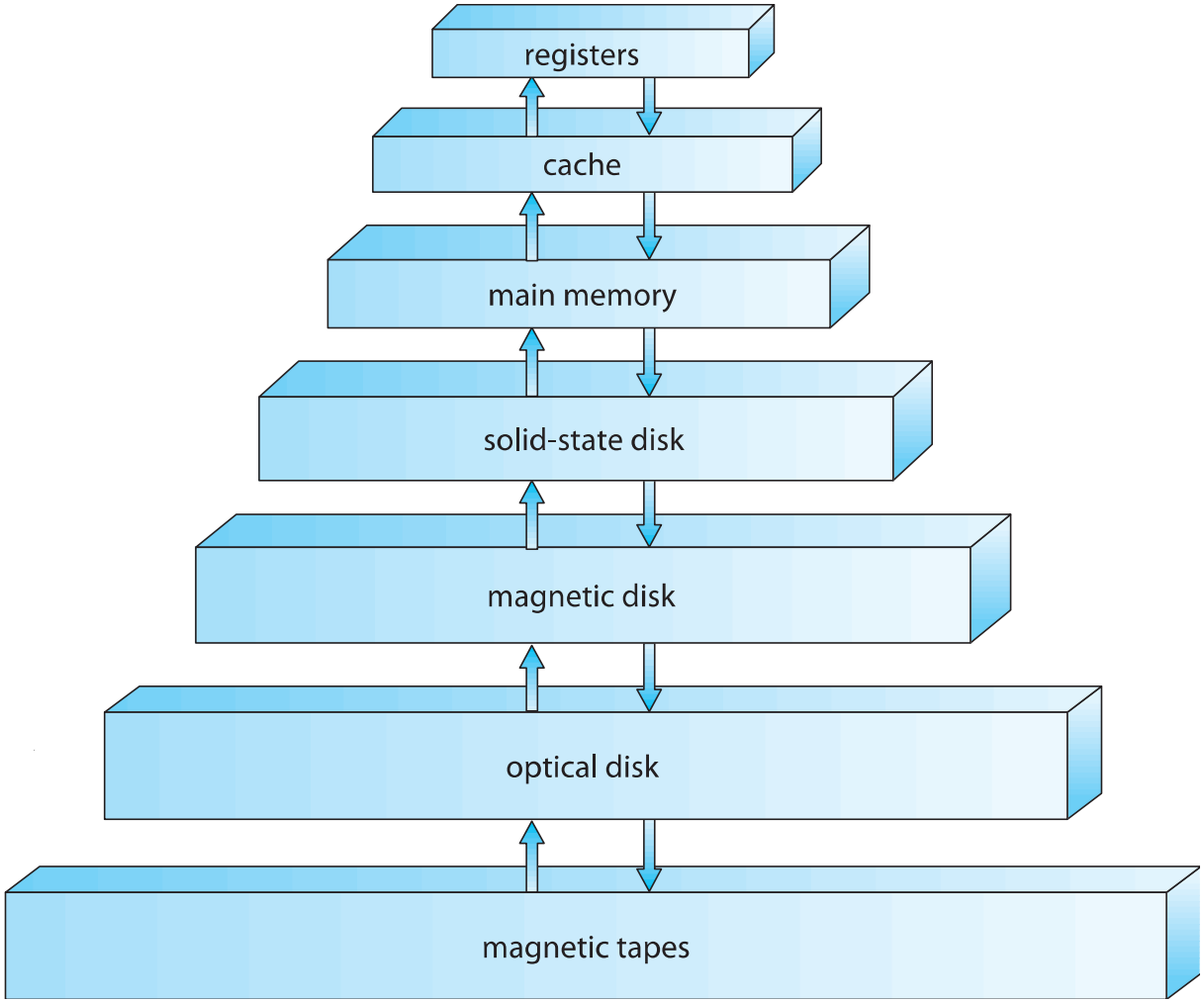
儲存體結構(Storage Structure)

- 程式和資料要先存放在主記憶體讓 CPU 存取才能執行。
- 主記憶體：CPU可以直接存取的儲存媒體
  - 通常是由隨機存取記憶體 (Random Access Memory，RAM) 所組成
  - 資料會隨著關機而消失，為揮發性儲存體 (Volatile Storage)。
- 輔助儲存體：主記憶體的延伸，可以提供大量的非揮發性儲存容量，屬於非揮發性。
  - 磁碟：覆蓋磁性記錄材料的硬性金屬或玻璃盤
    - 磁碟表面被邏輯分成，磁軌再被分成磁區(sectors)
  - 固態硬碟(solid-state disks)：比磁碟快、非揮發性
  - Flash memory

記憶體階層(Storage Hierarchy)

- 記憶體階層依下列因素以階層架構的方式組織，越高層的記憶體越快但越貴，越低層的記憶體則越慢但較便宜。
  - 速度(Speed)
  - 價格(Cost)
  - 可揮發性(Volatility)
- 快取技術(Caching)：基於程式執行時的區域性(Locality)，盡量將馬上要用到的資料拷貝到更快的儲存體，加快存取速度，例如主記憶體可視為輔助記憶體的快取記憶體(Cache)。
- Cache 的使用觀點就是基於程式的區域性 (Locality) 而設計的記憶單元

記憶體階層示意圖


registers
cache
main memory
solid-state disk
magnetic disk
optical disk
magnetic tapes

I/O 結構

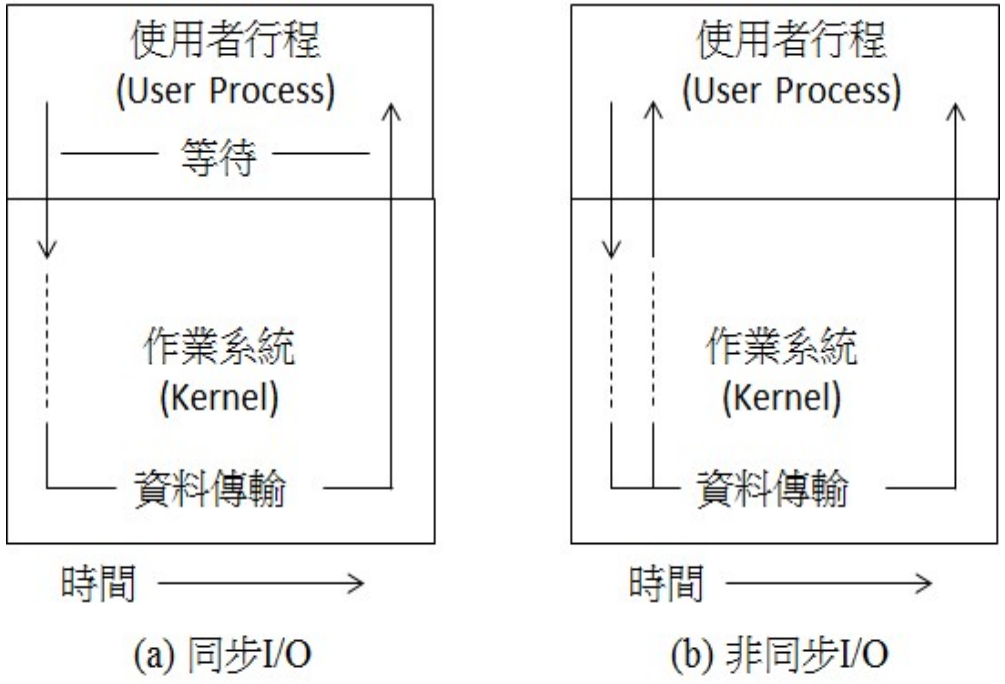
- 裝置驅動程式(device driver)：是一個允許高階電腦軟體與硬體互動的程式，提供一個硬體與軟體溝通的介面，經由主機板上的匯流排連接，使得硬體裝置上的資料交換成為可能。
- 當作業系統啟動一個 I/O 操作時
  - 裝置驅動程式會將 I/O 操作的命令放到裝置控制器的暫存器之中
  - 根據命令透過裝置控制器的區域緩衝區與裝置進行資料傳輸
  - 當傳輸完成，裝置控制器會觸發一個中斷來通知裝置驅動程式，將控制權轉交回OS，CPU 已完成該I/O。 CPU 就可以將裝置控制器之區域緩衝區的資料搬入主記憶體中，或從主記憶體搬到區域緩衝區。

I/O 結構(續)

- 三種將資料在I/O間傳送的方法有
  - Polling(輪詢)
    - CPU會周期性的檢查I/O裝置是否需要服務
  - Interrupt-driven I/O
    - 利用interrupt的機制，當一個I/O device需要服務時，會發出interrupt來通知CPU。
  - DMA(Direct Memory Access，直接記憶體存取)
    - 提供一個DMA controller，讓I/O device能夠直接在記憶體做資料的傳輸，不需要CPU的參與。

I/O 結構(續)

- 當 I/O 操作完成，基本上，CPU 執行權應該回到使用者的行程 (User Process)。然而，CPU 執行權回到使用者行程的情況有兩種返回方式：
  - 同步式 I/O (Synchronous I/O):使用者行程須等待I/O完成
  - 非同步式I/O (Asynchronous I/O):使用者行程不須等待I/O完成

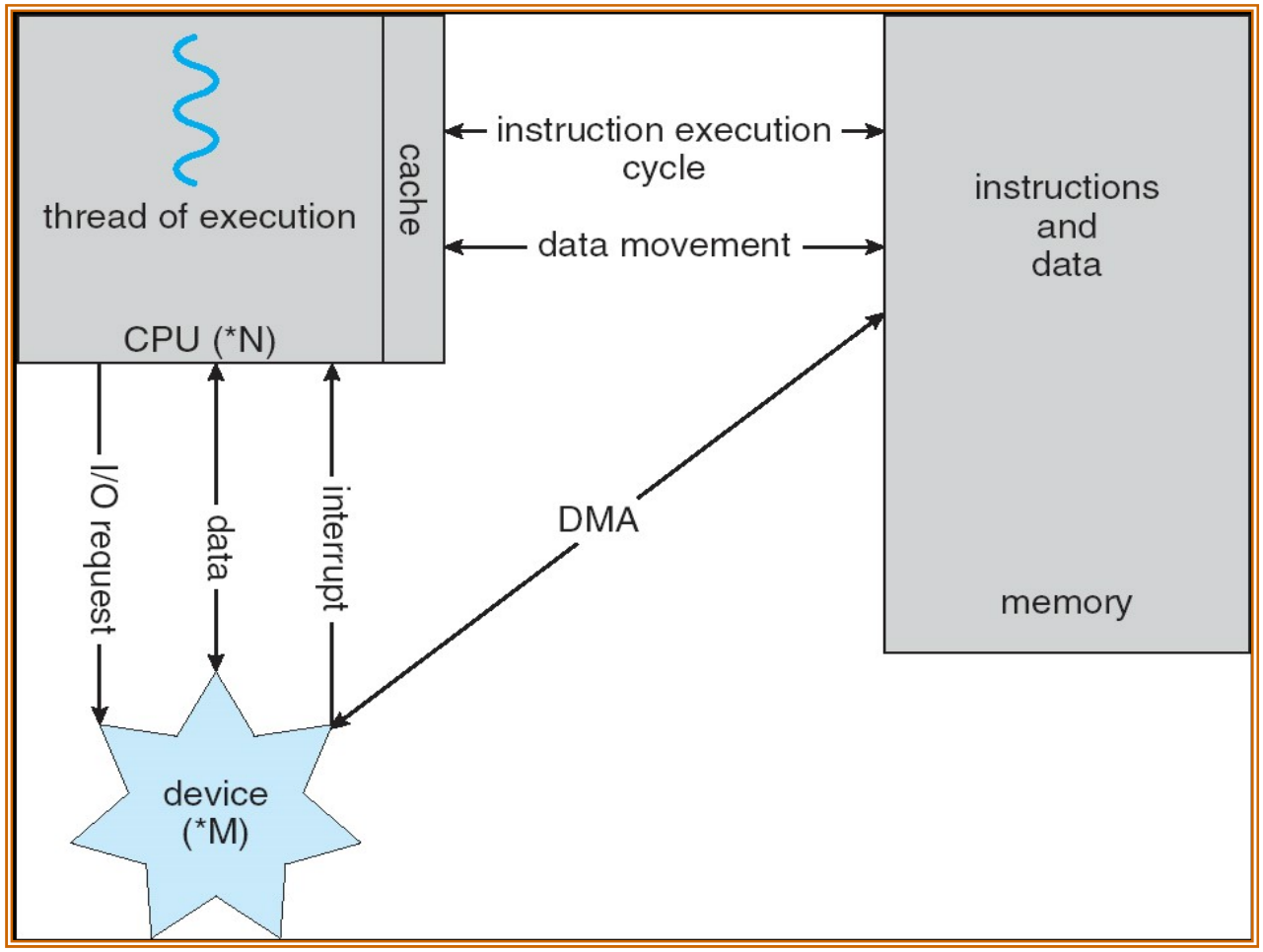


(a) 同步I/O　　(b) 非同步I/O

I/O 結構(續)

- 對資料傳輸速度較慢或傳輸量較小的周邊裝置(如滑鼠、鍵盤等)，可採取每傳送一個或幾個bytes即產生一次中斷的作法。
- 但對於需做高速大量傳輸的裝置(如disk I/O)，則須改用直接記憶體存取 (Direct Memory Acces，DMA)方式，才不會造成CPU太大負擔而影響系統效能。
- DMA作法
  - DMA 控制器會以不透過 CPU 介入方式直接將緩衝區 (Buffer) 之區塊資料 (Block Data) 傳到主記憶體。
  - 傳送一個資料區塊只產生一次中斷，而非每一位元組產生一次中斷。
  - DMA 傳送資料是利用時脈週期偷取方式(Cycle Stealing)，偷取CPU 的記憶體時脈週期 (Memory Cycle) 時間來傳送。

How a Modern Computer System Works



電腦系統架構(Computer-System Architecture)

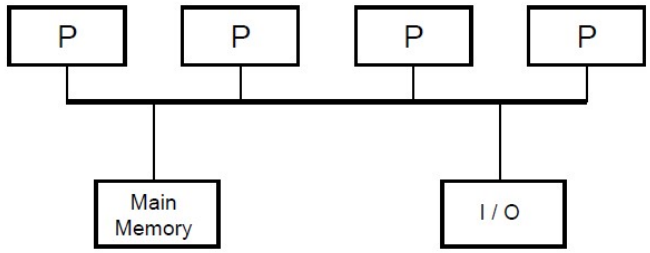
- 單一處理器系統
  - 一部機器中僅使用一個通用處理器及一些特殊用途的處理器
- 多處理器系統(multiprocessor systems)、平行系統(parallel systems)、多核心系統(multicore systems)
  - 一部機器中，同時具有多顆CPUs，但共用匯流排、記憶體及周邊裝置。受同一個clock控制，一般也受同一個OS所管控。
  - 優點包括:
    - 提升產能(Increased throughput)：但N 個處理器的加速倍數不是 N 倍，它會有一些額外負擔 (Overhead)。
    - 經濟效益(Economy of scale)：因為它們分享周邊設備、大容量儲存設備、電源等設備，所以成本較低。
    - 增加可靠度(Increased reliability)：如果有一個處理器故障，可以協調將原在故障處理器執行的工作安排給其它處理器來完成，系統不會因此無法繼續運作。

多處理器系統(續)

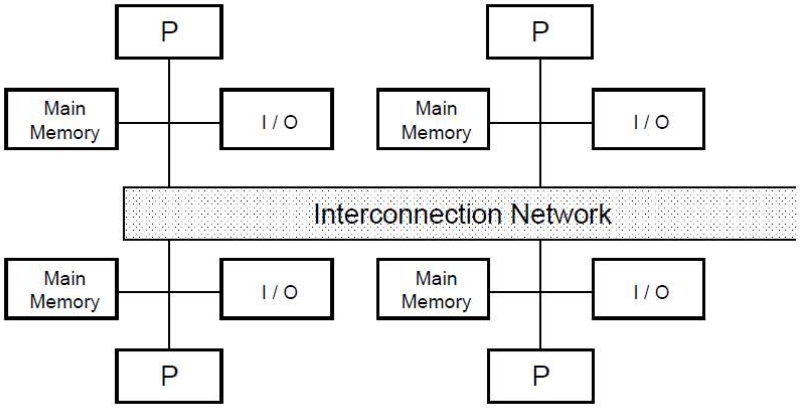
- 適度降級(graceful degradation)：繼續提供服務的能力與殘存硬體的程度呈正比的關係，稱適度降級。
- 具備容錯 (fault tolerant)能力：無視硬體失效而持續作業的能力
- 可分成對稱式多處理系統 (Symmetric Multiprocessing System) 和非對稱式多處理系統 (Asymmetric Multiprocessing System) 兩種。
- 多處理器系統可分成兩種型態:
  - 對稱式多處理器系統 (Symmetric Multiprocessing，SMP)
    - 所有處理器均扮演同等角色，無主僕 (Master-Slave) 關係。
  - 非對稱式多處理器系統 (Asymmetric multiprocessing)：
    - 處理器間具備主僕關係(boss-worker or master-slave relationship)
    - 有一個主處理器控制著系統，其它處理器則接受主處理器分配工作，或者執行預定好的工作。

多處理器系統(續)

- 多處理器系統記憶體存取模型可有兩種變化
  - 一致性記憶體存取 (uniform memory access，UMA)
    - 每一處理器對任一記憶體的存取時間都一樣快。

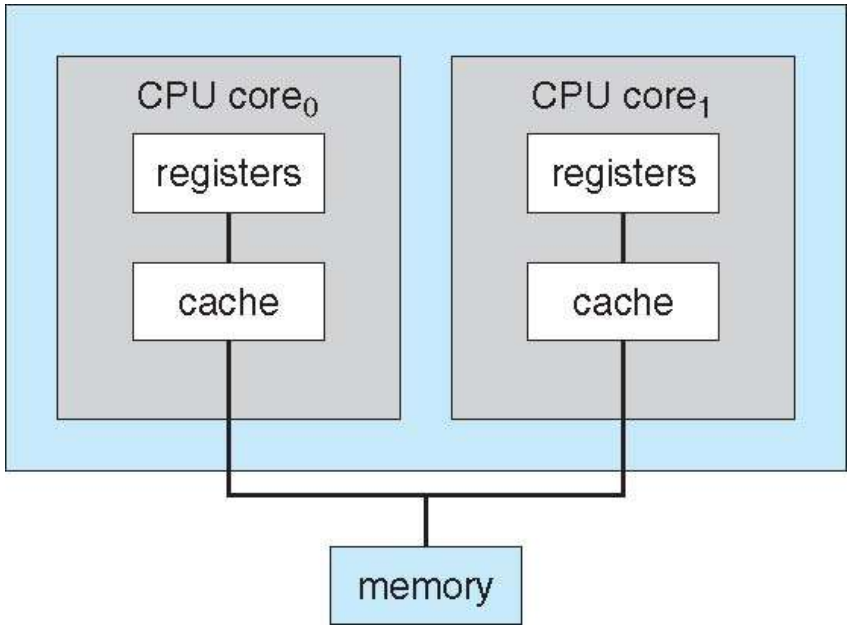


  - 非一致性記憶體存取(non-uniform UMA, NUMA)
    - 處理器對記憶體的存取時間有快有慢



多處理器系統(續)

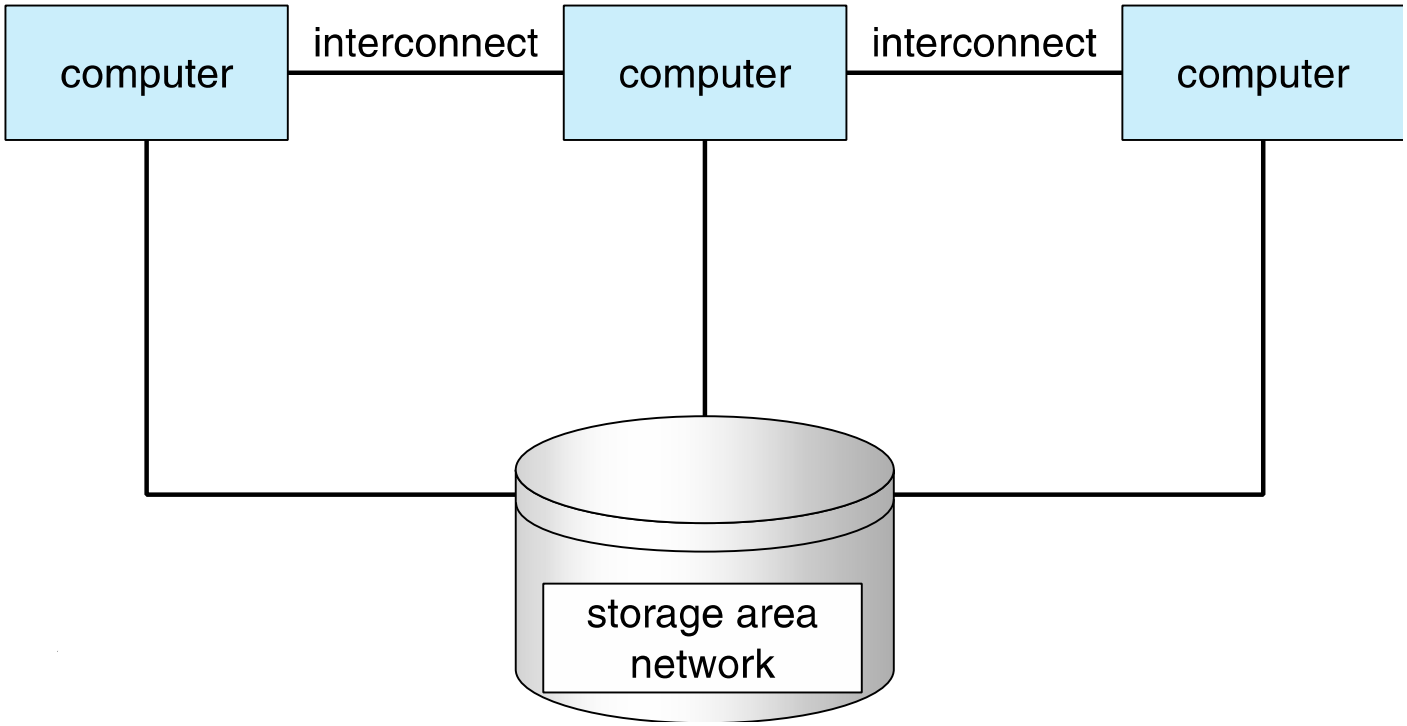
- 多核心系統(Multicore system)
  - 將多顆CPU封裝在同一晶片
  - 比多個單核心晶片更有效率，且較省電



叢集式系統(Clustered Systems)

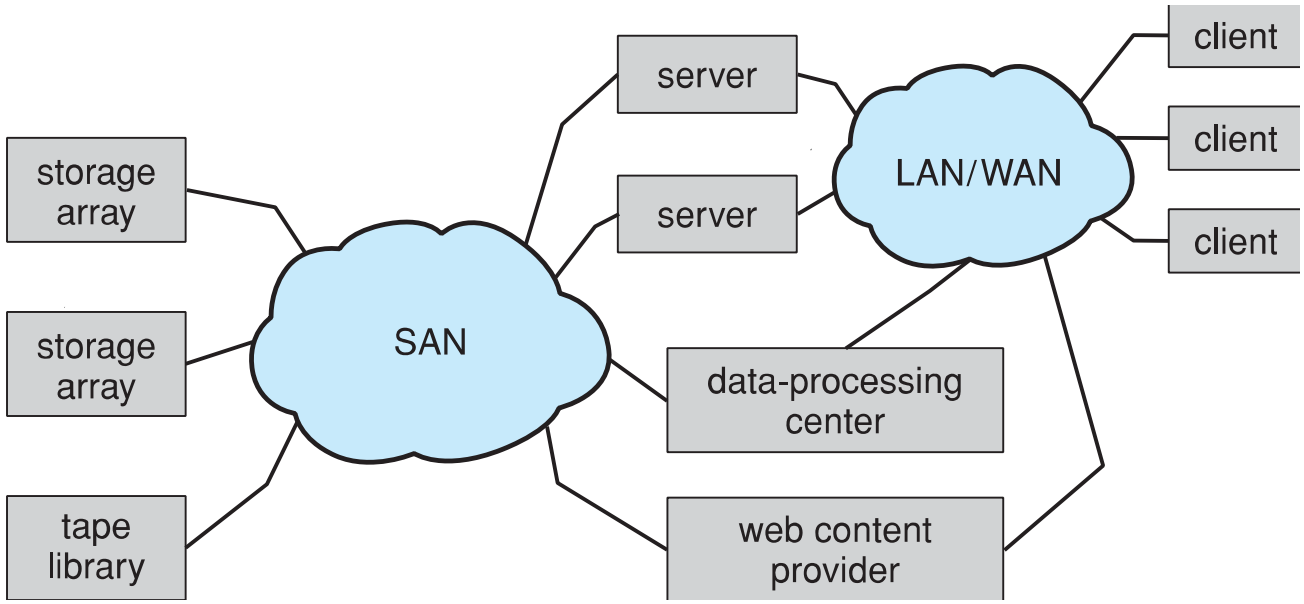
- 將多個個別系統透過網路串聯起來一起工作，又稱鬆耦合系統(loosely-coupled systems)
  - 通常經由儲存區域網路(storage-area network, SAN)共用儲存體
  - 提供高取得率(high-availability)的服務，某些系統失效後依然可以繼續提供服務
  - 可分成非對稱(asymmetric)叢集與對稱(symmetric)叢集對稱(symmetric)叢集
  - 可用來提供高性能運算(high-performance computing, HPC)
    - 應用程式必須以能平行處理(parallelization)的方式來撰寫
  - 為確保對分散式資源的存取保持一致性，叢集系統須具有分散式鎖定管理員(distributed lock manager, DLM) 的功能。

叢集式系統示意圖


computer
interconnect
computer
interconnect
computer
storage area
network

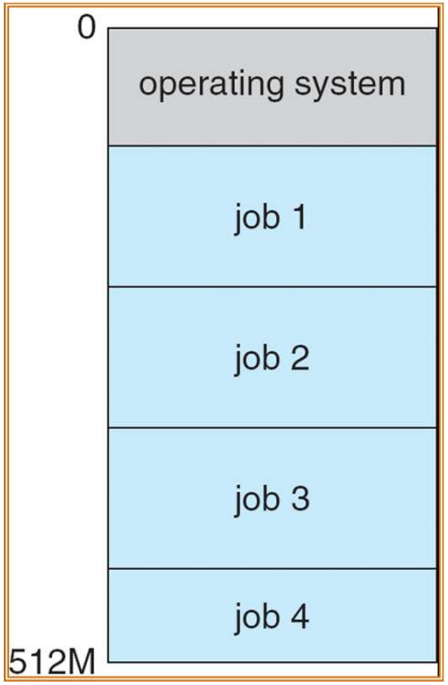
儲存區域網路(Storage Area Network)

- 一種透過網路的存儲結構，是以數據存儲為中心的。
- 採用可擴展的網路拓撲結構連接伺服器和存儲設備，並將數據的存儲和管理集中在相對獨立的專用網路中，透過伺服器提供數據存儲服務。
- 伺服器和存儲設備之間的多路、可選擇的數據交換消除了以往存儲結構在可擴展性和數據共用方面的局限性。



作業系統結構(Operating System Structure)

- 不同類型的作業系統其內部結構會有所不同
- 多元程式規劃系統(Multiprogramming System)
  - 目的是讓CPU使用率(CPU Utilization)極大化
  - 將多個工作(jobs) 載入到記憶體中，透過工作排班(job scheduling)挑選出一個工作來執行，當正在執行的工作需等待的時候(如等待I/O完成)，OS再挑選其他工作來執行，使得CPU一直有一個工作可執行。



作業系統結構(續)

- 分時系統(Time-Sharing System or Multitasking System)
  - 是多元程式規劃作法的延伸
  - 可同時支援多使用者及多工作業
  - 藉由快速交替切換多個工作的執行，提供使用者與電腦互動的能力。要讓使用者與電腦互動很順暢，系統回應時間 (Response Time) 要夠短(< 1 second)。
  - 作法是將CPU的時間切割成很多小時間片段(time quantum)，CPU在各行程或使用者間不停地切換執行。頻繁的切換使每個使用者都以為自己程式一直在執行。
  - 需用到虛擬記憶體、檔案系統、磁碟管理、保護及同步機制。

作業系統結構(續)

- 多元程式規劃和分時系統的主要差異是切換工作的時機不同
  - 多元程式規劃要做等待時切換(類似自動放棄CPU的使用權)
  - 分時作法是CPU時間額度用完了(類似被迫放棄CPU的使用權)

作業系統的運作(Operating-System Operations)

- 現代作業系統多為中斷驅動
- 事件(Events) 透過中斷方式來產生
- 為了確保系統正常運作，一個錯誤的程式的執行不可造成影響到其他程式的執行，在硬體與作業系統上需有一些保護機制的搭配。
  - 雙模式(Dual-mode)及多模式(Multimode)的支援
  - CPU保護-透過計時器(timer)
  - 記憶體保護
  - I/O保護
  - 系統呼叫(system calls)

雙模式(Dual-mode)

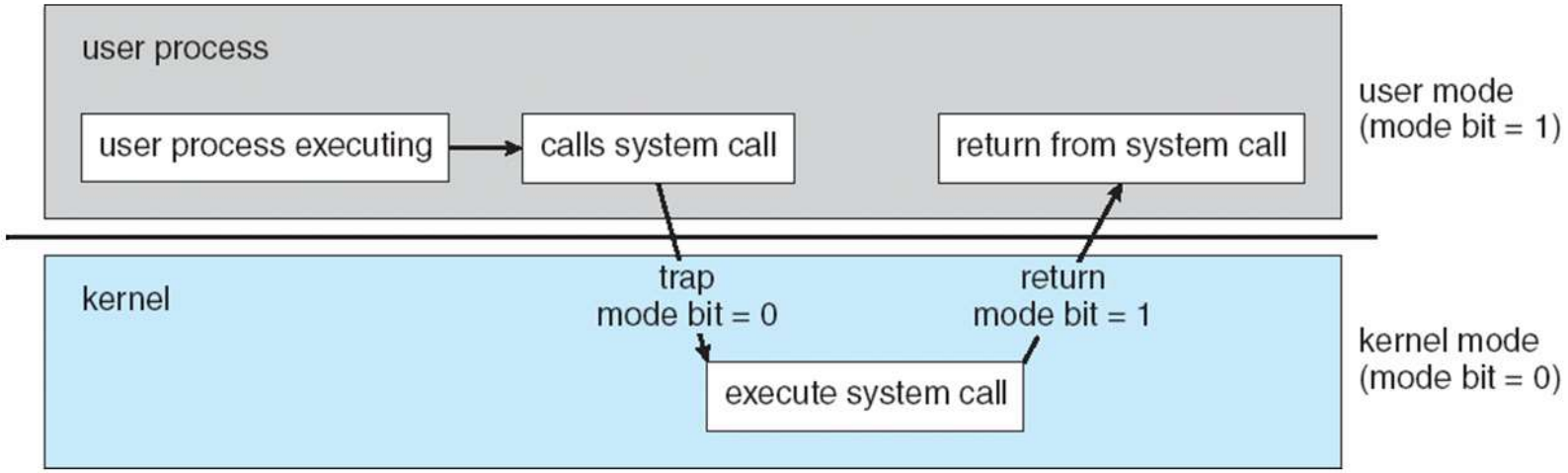
- 分成使用者模式 (User Mode) 與核心模式(Kernel mode，也可稱為 Supervisor Mode 、System Mode或 Priviledged Mode)
- 利用硬體上的模式位元 (Mode Bit) 進行模式區別
- CPU指令集內的特權指令(priviledged instructions)只能在核心模式下執行，無法在使用者模式下執行。
- 在使用者模式下執行特權指令，會發生軟體中斷 (Software Interrupt, 或稱陷阱: Trap)，CPU 控制權會交回給作業系統。
- 當使用者行程進行系統呼叫(system call)時，會改變為核心模式，從呼叫返回時則重置為使用者模式。
- 目前亦有很多CPU支援多模式，例如增加虛擬機器管理員模式(VMM)模式

CPU保護-透過計時器(timer)

- OS透過使用計時器可避免行程因進入無窮迴圈而無法結束或是霸占資源的情形
- 計時器是一個硬體機制，依計時設定在一定的時間後產生中斷。
- 當計時器中斷發生時，作業系統可取回控制權來決定要進行何種處理。
- 計時器還可用在分時系統之執行時間額度控制之用途。

系統呼叫(System Calls)

- 系統呼叫指運行在使用者行程向作業系統核心請求需要更高權限運行的服務，如設備IO操作或者行程間通信等。
- 系統呼叫處理過程



作業系統四大主要工作

- 行程管理(Process Management)
- 主記憶體管理(Memory Management)
- 儲存體管理(Storage Management)
- 保護與安全(Protection and Security)

行程管理簡介

- 一個執行中的程式稱為行程 (Process)。程式是靜態被動實體(passive entity)，而行程是動態主動實體(active entity)。
- 行程需要資源來完成其任務，如:CPU、記憶體、I/O、檔案及初始資料
- 行程結束時需要收回任何可以重新使用的資源
- 一個行程是利用 CPU 的程式計數器 (Program Counter)來指定下一個執行指令的位址，來導引程式的執行順序。
- 通常系統會有許多行程，有些作業系統讓許多行程在一個或多個CPU上並行地執行(concurrently running)，藉由讓CPU在許多行程/執行緒(thread)間輪流地執行來完成並行執行。

行程管理簡介(續)

- 行程管理的主要功能：
  - 負責建立或刪除使用者行程或系統行程
  - 要能暫停執行(Suspending)或恢復執行 (Resuming)行程
  - 提供多個行程間之行程同步 (Process Synchronization) 機制
  - 提供行程之間之行程通訊 (Process Communication) 機制
  - 提供行程之死結處理 (Deadlock Handling) 機制

主記憶體管理(Memory Management)

- 程式執行時所需的指令及資料都須先搬入主記憶體中
- 主記憶體可以被 CPU 和 I/O 裝置分享使用。 CPU 在指令讀取週期 (instruction-fetch cycle) 自主記憶體讀取指令，在資料讀取週期 (data-fetch cycle) 自主記憶體讀取或寫入資料。
- 記憶體管理的主要功能
  - 追蹤主記憶體的使用情形，哪些部分已被使用了及被誰使用
  - 決定那些行程(或部分的行程)和資料要移入或移出記憶體空間
  - 視需要分配 (allocating) 與回收 (deallocating) 主記憶體空間
- 虛擬記憶體(virtual memory)管理藉由輔助記憶體的支援允許行程之位址空間大於實際主記憶體的大小的記憶體空間，也可以提供更多行程同時載入主記憶體中執行，可以提高系統產能。

儲存體管理(Storage Management)

- 儲存體管理可分成三大單元
  - 檔案系統管理(File-system management)
  - 輔助儲存體管理(Mass-storage management)
  - I/O子系統管理(I/O subsystem management)

檔案系統管理(File-System Management)

- 作業系統提供使用者一個一致性的資料儲存的邏輯觀點，使得使用者不會因在不同物理特性的儲存裝置上進行資料儲存作業而感到不便。
- 檔案(File)即為該邏輯觀點的儲存單元。
- 將儲存裝置的物理特性抽象化來定義檔案
- 在一個多人使用的環境，作業系統也提供檔案的使用權限管理，確保檔案的安全
- 檔案系統管理功能：
  - 負責建立或刪除檔案
  - 負責建立或刪除目錄
  - 支援檔案和目錄的基本操作
  - 將檔案對映 (mapping)到輔助記憶體
  - 備份檔案到永久性儲存媒體中

輔助儲存體管理(Mass-Storage Management)

- 輔助儲存體用來儲存無法放入主記憶體或是必須長期保存的資料
- 輔助記憶體經常被使用，必須適當的管理才能發揮效益。
- 電腦的整體操作速度取決於磁碟速度與處理的演算法
- 磁碟管理的主要
  - 可用空間管理(Free-space management)
  - 儲存體配置(Storage allocation)
  - 磁碟排班(Disk scheduling)

各類儲存體的效能

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

I/O子系統 (I/O Subsystem)

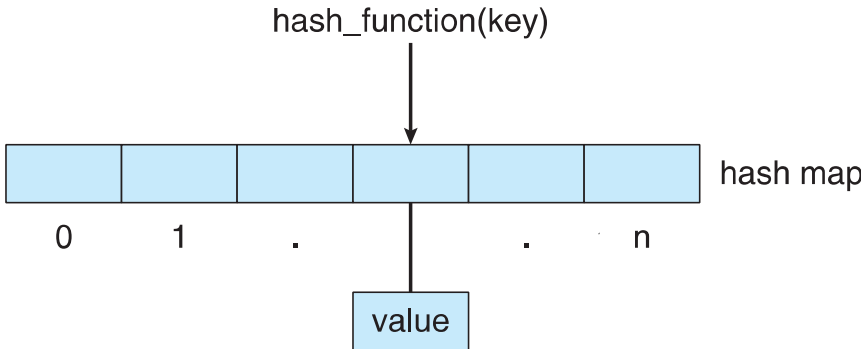
- I/O 子系統包括
  - I/O 所需之記憶體管理單元，包含緩衝區處理 (Buffering)、快取處理 (Caching) 、及線上同時周邊作業處理 (Simultaneous Peripheral Operation On-Line，簡稱 Spooling)
    - 緩衝區處理 (Buffering) : 負責暫時儲存被傳輸的資料
    - 快取處理 (Caching) : 負責將部分資料儲存在更快速的儲存體，以提升性能。
    - Spooling :將磁碟當作一個巨大的 Buffer 在使用，將慢速周邊裝置的輸出入資料先存放於快速周邊裝置(如硬碟)，然後才由CPU讀取，由於CPU不必和這些慢速裝置直接溝通，所以提升了CPU操作速度。
  - 通用的裝置驅動程式(device driver)介面
  - 特定硬體裝置驅動程式

保護和安全(Protection and Security)

- 保護(Protection) – 控制行程或使用者對於資源存取的機制
- 安全(Security) –防護系統免於外部與內部的攻擊
  - 包括拒絕服務(denial-of-service)、蠕蟲(worms)、病毒(viruses)、身份盜用(identity theft)、盜用服務(theft of service)等
- 系統通常先辨識使用者，來決定誰可以作什麼
  - 使用者識別碼(user ID)，一個使用者一個ID且都不同，決定使用者權限進行存取控制(access control)。
  - 群組識別碼(group ID) 允許一組使用者的權限被定義並控制管理
  - 某些時候，可透過權限提升(Privilege escalation) 方式允許使用者權限暫時獲得提升已進行某些作業。

核心資料結構(Kernel Data Structures)

- 在OS的實現上，會用到的資料結構有：
  - Lists, Stacks, Queues
  - Trees
  - 雜湊函數(Hash functions) and Maps



  - Bitmap ：一串 n 位元二進位數字來表示 n 個項目的狀態
- Linux核心的資料結構定義在以下包含檔案中<linux/list.h>, <linux/kfifo.h>, <linux/rbtree.h>

運算環境(Computing Environments)介紹

- 各種不同型態的運算環境介紹
  - 傳統運算(Traditional Computing)
  - 行動運算(Mobile Computing)
  - 分散式系統(Distributed Systems)
  - 主從式運算(Client-Server Computing)
  - 點對點(對等式)運算(Peer-to-Peer Computing)
  - 虛擬化 (Virtualization)
  - 雲端運算(Cloud Computing)
  - 即時嵌入式系統(Real-Time Embedded Systems)

傳統運算(Traditional Computing)

- 由於網路的普及與進步，使得傳統運算環境易發生很大的改變，界限也變模糊了。例如:
  - 公司建置入口網站(portals)，提供Web方式來存取內部資源。
  - 網路電腦(Network computers, thin clients) 的安全性與易維護性，也逐漸取代傳統工作站。
  - 一般家用PC也可當成各種伺服器來提供網路服務

行動運算(Mobile Computing)

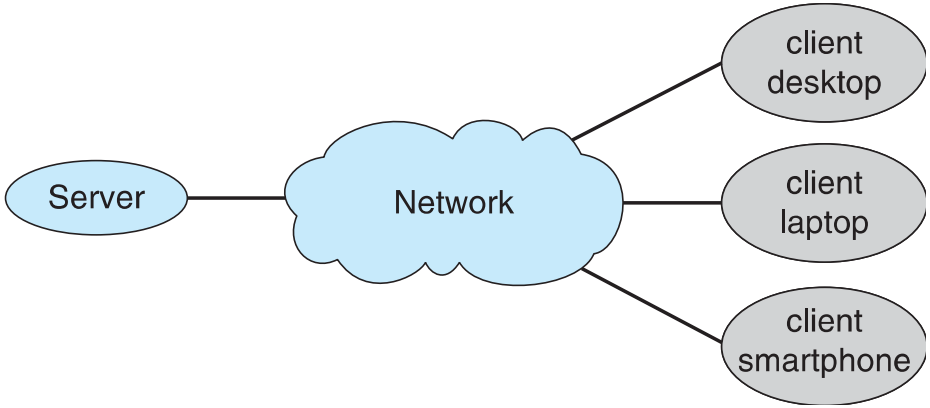
- 智慧型手機與平板電腦上的運算
- 體積小，重量輕
- 並具備許多感測器(GPS、陀螺儀、加速規)
- 許多新型態的apps及應用
- 使用無線或電信網絡作連接
- 兩大OS領導者是 Apple iOS 和 Google Android

分散式系統(Distributed Systems)

- 一組彼此獨立的系統(可能是異質型態的系統)以網路連結在一起
  - 網路是通信路徑, TCP/IP則是最普遍的網路協定
  - 網路依連線範圍，可分成
    - 區域網路(Local Area Network, LAN)
    - 廣域網路(Wide Area Network, WAN)
    - 都會網路(Metropolitan Area Network, MAN)
    - 個人區域網路(Personal Area Network, PAN)
- 分散式系統之作業系統類型
  - 網路作業系統：連接在一起的電腦，可以使用不同的作業系統，這些電腦彼此共用資源時，必須使用遠端登錄或直接以檔案傳輸方式傳輸檔案使用。
  - 分散式作業系統：連接在一起的電腦，使用相同的作業系統，任何遠端的資源，都被作業系統視為本身的資源，而可以直接存取。

主從式運算(Client-Server Computing)

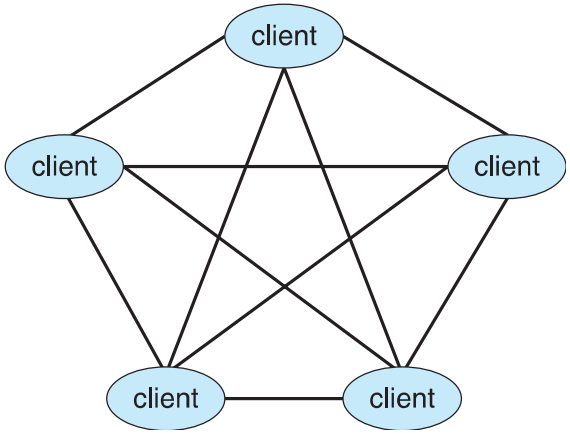
- 伺服器通過網路對外提供服務，由伺服器回應客戶端的請求



- 伺服器系統類型
  - 運算伺服器系統(Compute-server system)
    - 提供介面給客戶端，讓客戶端可送出服務請求(例如，資料庫伺服器提供資料庫的事務處理和數據查詢)
  - 檔案伺服器系統(File-server system)
    - 提供檔案系統介面給客戶端，讓客戶端可在伺服器進行檔案的處理。
  - 列印伺服器系統(Printer-server system)
  - …

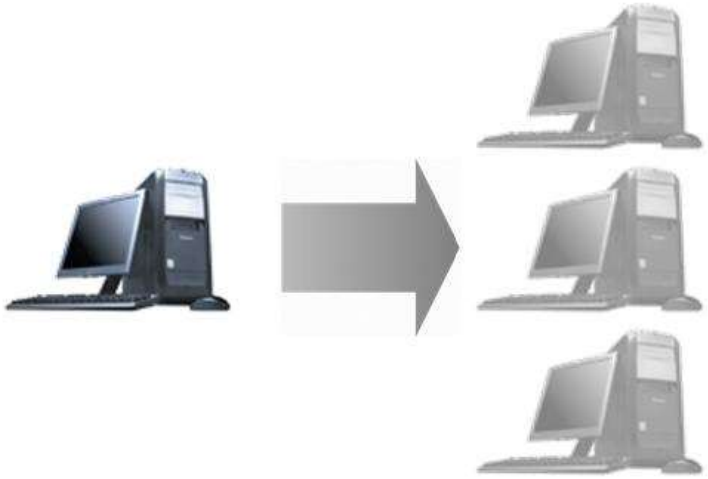
點對點(對等式)運算(Peer-to-Peer Computing)

- 另一種模式的分散式系統
- 由節點(nodes)組成，所有節點的身份都是對等的，每一個節點可以伴演客戶端、伺服器或是兩者皆可。
- 節點須主動加入(join)後方可提供或請求服務
- 服務查詢方式
  - 可透過集中註冊服務方式讓其他節點知道
    - 如Napster所提供的音樂共享服務
  - 亦可透過廣播服務要求並經由搜尋協定(discovery protocol)回應服務要求
    - 如Gnutella所提供的檔案分享
- Skype採混合式對等式運算架構，利用一個端點系統來負責管理。



虛擬化 (Virtualization)

- 虛擬化（Virtualization）[Wiki解說]：是一種資源管理技術，將計算機的各種實體資源，予以抽象、轉換後呈現出來並可供分割、組合為一個或多個電腦組態環境。一般所指的虛擬化資源包括計算能力和資料儲存。



虛擬化 (Virtualization)

- 術語介紹
  - 寄宿機 (Host Machine)即虛擬機管理程序所在的主機系統
  - 客戶機 (Guest Machine)即運行在虛擬化管理器之上的虛擬機系統
  - 虛擬機管理程序VMM (Virtual Machine Manager) 可以監視虛擬機的運作
- 開機的作業系統稱為主機作業系統 (Host OS)。
- 架在虛擬機管理程序之上是虛擬機（Virtual Machine），其內部運行的是客戶機作業系統(Guest OS) 。
- VM都是一些相互隔離的客戶機作業系統，它們將底層硬體平台視為自己所擁有。但是實際上，是虛擬機管理程序為它們製造了這種假象(虛擬化)。

虛擬化 (Virtualization)(續)

- 使用虛擬主機的好處:
  - 虛擬主機並不互相干擾， 藉由保護系統資源，虛擬主機可以供更好的安全性。
  - 讓系統在不用停機而發展工作仍可完成。
  - 藉由虛擬主機的普及化，可解決系統相容性 (Compatibility) 問題。
- 虛擬化方法
  - 基本上虛擬化解決方案是要進行將實體機器虛擬化，這台機器可能直接支持虛擬化，也可能不會直接支持虛擬化。
  - 若硬體不會直接支持虛擬化，那麼就需要使用虛擬化管理程序層的支持虛擬化。虛擬機管理程序稱為VMM，可以看作是平台上硬體及作業系統的抽象化。

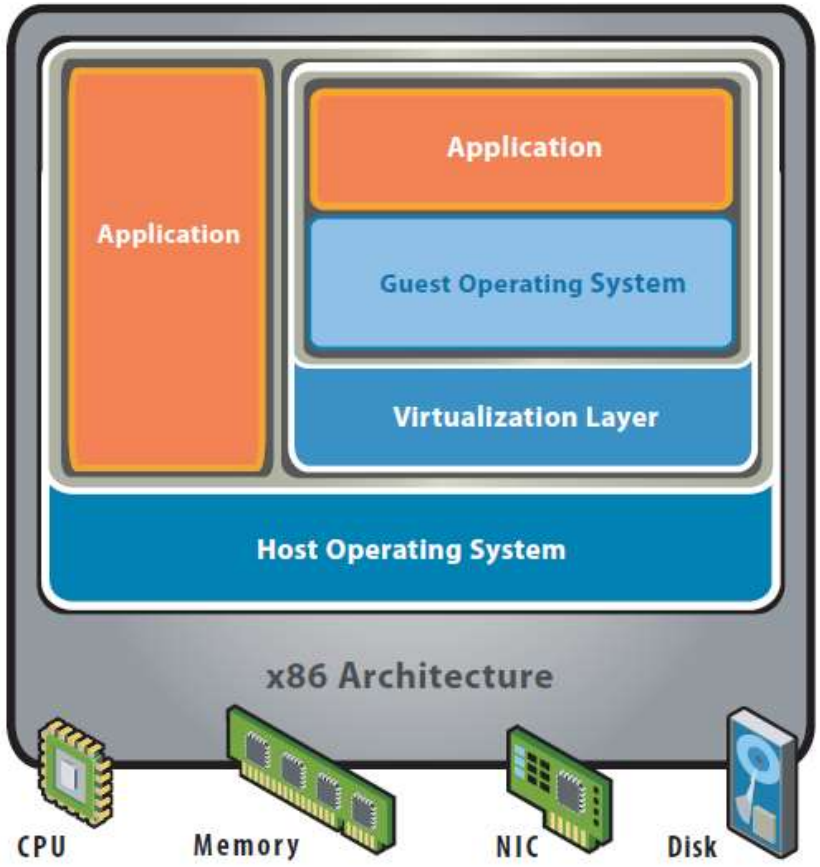
虛擬化 (Virtualization)(續)

- 不同工作性質的虛擬機
  - Process VM:
    - 只虛擬一個Process 的行為，可能是為了支援舊架構上的應用程式， e.g., Apple Rosetta；
    - 也可能是為了跨平台上執行應用程式e.g., Java 。
  - System VM:
    - 虛擬的是一個完整的OS，需要更複雜的處理，如：VMware ESX 和 Citrix XenServer。

虛擬化 (Virtualization)(續)

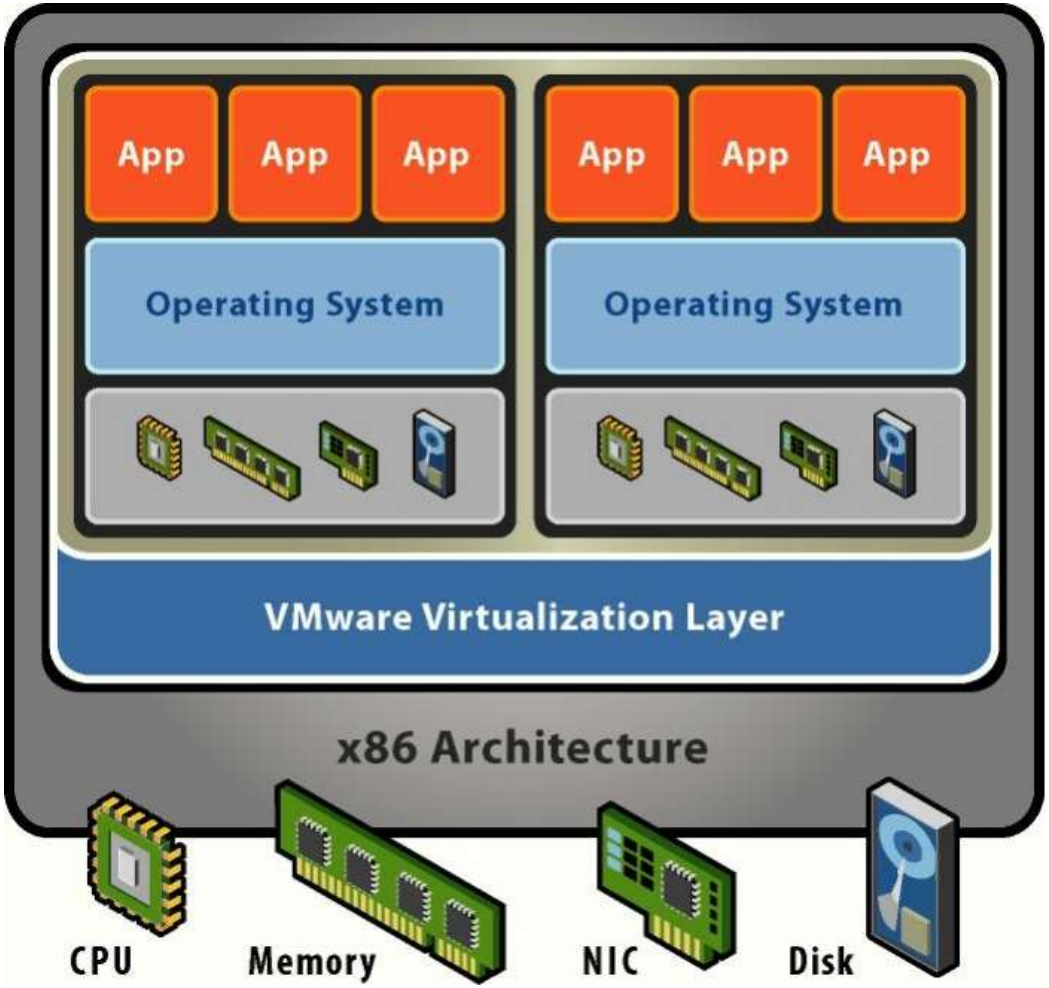
- System VM 又區分為Hosted Architecture及Bare Metal Architecture兩種架構
  - Hosted Architecture
    - 將VMM 建立在一個Native OS上。
    - 如：Virtual PC, Virtual Box, VMware Workstation 7, KVM, OpenNebula, OpenStack, and etc.。
  - Bare Metal Architecture
    - 直接將VMM (或Hypervisor)建立在硬體之上。
    - 如： VMware ESX, Hyper-V R2, Citrix XenServer, Xen, and KVM(後期)。

VMware Example



Hosted Architecture
(source: VMware)

VMware ESX/ESXi Example



Bare Metal Architecture
(source: VMware)

虛擬化 (Virtualization)(續)

- Virtualization Hardware Support技術
  - Intel CPU 提出了Intel-VT 技術。
  - AMD CPU提出了AMD-V技術。
  - ARM Cortex A15 也將支援。

雲端運算(Cloud Computing)

- 經由網路提供運算、儲存、甚至是應用程式作為服務
- 通常透過虛擬化 (Virtualization)與分散式運算技術實現，可視為虛擬化技術的邏輯延伸
- 雲端運算之三種服務模式:
  - 軟體即服務 (Software as a Service， SaaS)
  - 平台即服務 (Platform as a Service， PaaS)
  - 基礎架構即服務 (Infrastructure as a Service， IaaS)

	IaaS 基礎架構即服務	SaaS 軟體即服務	PaaS 平台即服務
提供服務	基礎建設	軟體	平台
服務項目	伺服器 網路頻寬 硬體管理	各種線上應用軟體	提供軟體開發、 測試的環境
服務對象	IT管理人員	終端用戶	軟體開發人員
服務提供者	- Oracle Compute Cloud - IBM CloudBrust - Amazon EC2	- iCloud - Google Apps - Office 365	- Google App Engine - Salesforce.com - Microsoft Azure - Amazon EC2

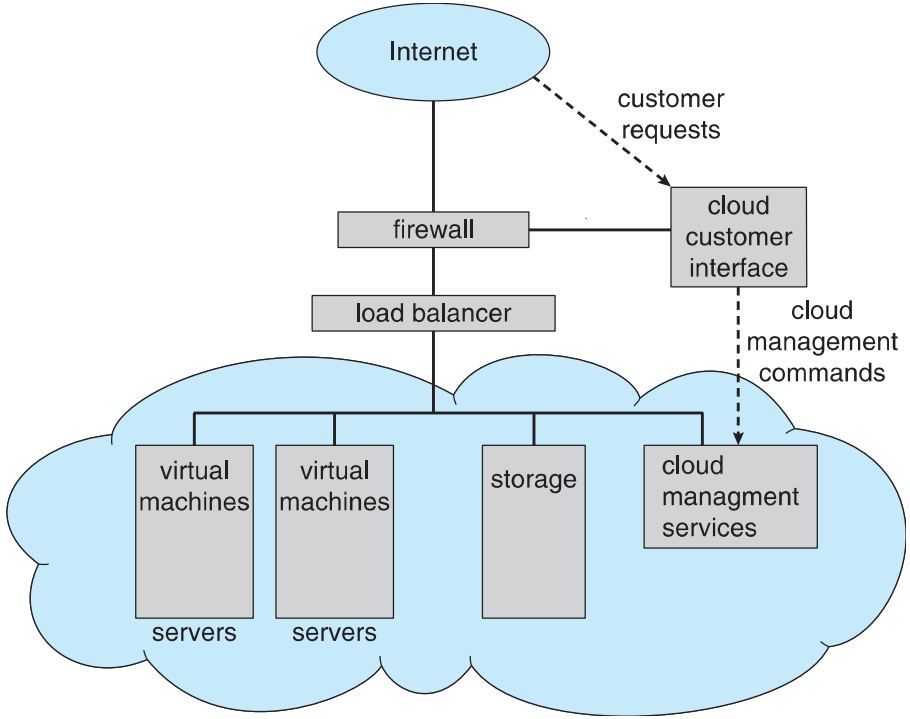
資料來源：原文發表於寫點科普https://www.inside.com.tw/2017/06/27/cloud-computing

雲端運算(Cloud Computing)(續)

- 雲端運算的佈署模式可分為
  - 公用雲(Public cloud)：經由網際網路給任何願意付費者取得的雲
  - 私有雲 (Private cloud) ：一個企業或組織所提供其內部營運與管理之雲
  - 混合雲(Hybrid cloud) ：私有雲與公有雲的混合

雲端運算(Cloud Computing)(續)

- 雲端運算環境由傳統OS、VMM(Virtual Machine Manager)和雲端管理工具組成
- 如下圖，一個公用雲提供IaaS服務的情形
  - 透過防火牆來保護雲端所提供的服務與雲端客戶介面
  - 負載平衡器多個應用程式的網路流量進行分配



即時嵌入式系統 (Real-Time Embedded Systems)

- 嵌入系統是最普遍的電腦應用形式
- 即時系統需要有預設的時間限制 (Well-defined Time Constraints)。即時系統處理工作必須在預設的時間限制內完成，否則就是失敗。
- 即時作業系統有嚴格的處理時間限制
  - 必須在時間限制內完成處理
  - 只有符合時間限制才被認為正常運作
- 即時系統分成兩大類
  - 硬即時系統 (Hard Real-Time System)
    - 保證關鍵性工作 (Critical Tasks) 須準時完成， 故系統的所有工作的延遲時間(Delay) 都必須受到限制。
  - 軟即時系統 (Soft Real-Time System)
    - 對於關鍵性工作給予較高的優先權 (Priority)，但不保證一定會準時完成。

開放原始碼作業系統
(Open-Source Operating Systems)

- 可以取得原始碼格式的作業系統
- 由倡導“copyleft”和GNU Public License (GPL)的自由軟體基金會(Free Software Foundation，FSF)發起
- 如GNU/Linux、BSD UNIX (包含Mac OS X的核心)、Solaris等