2024 年秋季

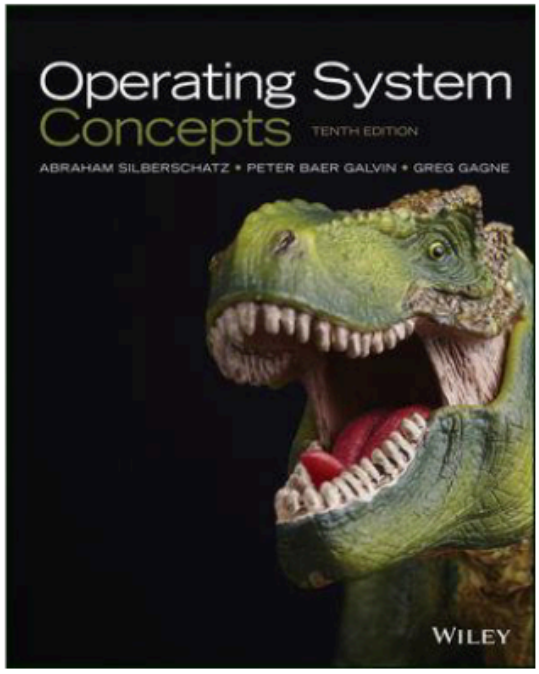
COMP 3511 操作系统

讲座和实验室/教程

- 讲座（2024年9月2日至11月29日）：
  - L1 周一、周三 9:00AM - 10:20AM，LT-F 电梯 25-26 L2 周二、周四 3:00PM - 4:20PM，LT-G，电梯 25-26 L3 周二、周四 9:00AM - 10:20AM , G010 CYT大厦
- 实验室教程
  - LA1 周五 10:30AM - 12:20PM，LT-G LA2 周一 03:00PM - 04:50PM，G010，CYT 大楼 LA3 周四 06:00PM - 07:50PM，LT-C
- Course Website: https://course.cse.ust.hk/comp3511/
- Instructors: Junxue Zhang (L1), Bo Li (L2) and Mo LI (L3)

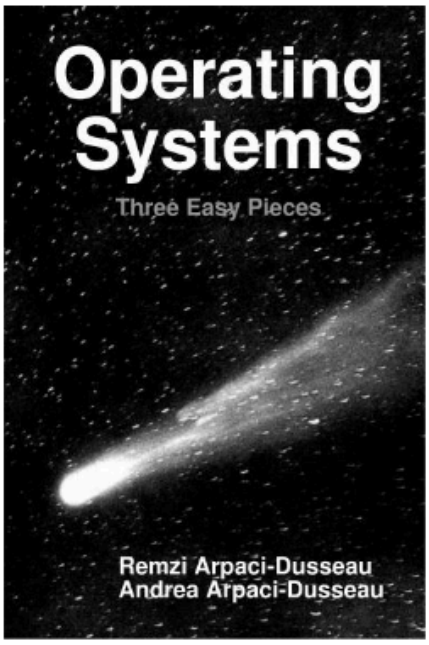
教科书

- 操作系统概念，A. Silberschatz、PB Galvin 和 G. Gagne，第 10 版



参考书

- 操作系统：三个简单的部分，Remzi H. Arpaci-Dusseau 和 Andrea C. Arpaci-Dusseau
- 在线（免费访问）：http://pages.cs.wisc.edu/~remzi/OSTEP/#book-chapters



课程先决条件

- COMP 2611 或 ELEC 2300 或 ELEC 2350（计算机组织）计算机组织 – 冯·诺依曼机、CPU、流水线、缓存、内存层次结构、I/O 系统、中断、存储和硬盘驱动器

- COMP 2011 或 COMP 2012H（C 编程）
  - UNIX/Linux 基本编程要求 - C 编程
  -

实验室和教程

- 9 实验和教程 – 暂定时间表以讲座进度为准
  - 第 1 周无实验 实验#1（第 2 周）：Linux 简
  - 介 实验#2（第 3 周）：C/C++ 编程 实验#3
  - （第 4 周）：Linux 进程、pipe() 和项目#1
  - 实验#4 （第 5 周）：回顾
  - 实验 #5（第 6 周）：项目 #2 实
  - 验 #6（第 7 周）：回顾 第 8 周
  - （期中周）没有实验 实验 #7
  - （第 9 周）：回顾
  - 实验#8（第 10 周）：项目#3
  - 实验#9（第 11 周）第 12 周
  - 回顾缓冲周

评分方案

- 4 项家庭作业 - 书面作业 – 20%（每项 5%）
  - HW #1（第 2-4 周）
  - HW #2（第 5-7 周）
  - HW #3（第 8-10 周）
  - HW #4（第 11-13 周）
- 3 个项目 - 编程作业 – 30%
  - 项目 #1（第 4-6 周）(10%)
  - 项目 #2（第 7-9 周）(10%)
  - 项目 #3（第 10 -12 周）(10%)
- 期中考试（~#8/9 周）- 20%
- 期末考试 - 30%

剽窃政策

- 协作、讨论和文案是有区别的！
- 第一次：所有涉及者得零分，并会报告给ARR 第二次：需要终止（不及格等级）
- 期中或期末考试中的任何作弊都会导致自动不及格等级

讲座形式

- 讲座：
  - 讲义在讲课前提供
- 教程和实验室
  - Unix环境、编辑器（vim）、编译和运行程序、Makefile C++和C
  - 编程基础
  - 编程作业教程 C 编程 API 和接口 通过更多示例和练
  - 习补充材料
  -
- 阅读教材和参考书中相应的材料
  - 讲义没有也不可能涵盖所有内容
- 章节摘要
  - 每章末尾都有全面的总结

作业

- 书面作业
  - 在规定的时间内截止 如果对成绩有任何争议，请联系相应的助教 重新评分请求将在作业成绩公布后两周内获得批准
  - 延迟政策：减少10%，仅允许延迟一天
- 编程作业 - 个人项目
  - 在指定时间截止 在 CS Lab 2 Linux 机器上运行 使用 Canvas 提交
  - 成绩发布后两周内会批准重新评分请求
  - 延迟政策：减少10%，仅允许延迟一天

期中考试和期末考试

- 期中考试
  - 时间：10月25日（星期五（第8周）下午6:30 –
  - 晚上8:30 地点：CYT LT-L、LT-B、LT-C
- 期末考试
  - 待定
- 所有考试均为开卷和开卷（硬拷贝）
  - 不允许使用任何电子设备
- 除非有以下情况，否则不会进行补考
  - 特殊情况，如生病，需提供证明信，考前须告知导师

学习技巧

- 参加讲座和实验室教程
  - 讲座前下载讲座/实验笔记通过示例解释重要概念
  -
- 独立完成作业和项目
- 这是为了测试您的知识以及您的理解程度每周花 30 分钟左右复习内容
  - 章节总结有帮助 这可以为您以后准备考试时节省大量
  - 时间 您不能指望在考试前 2-3 天学习所有内容 知识是
  - 逐步积累的
  -
- 尽早开始你的项目
- 制定项目计划 在讲座期间或之后提出问题！
  - 不要把提问拖到临近考试的时候

你应该学习什么

- 定义操作系统设计和实现中使用的基本原理、策略和算法分析和评估操作系统功能
-
- 了解操作系统内核的基本结构，并识别各个子系统之间的关系
- 识别指示潜在操作系统问题的典型事件、警报和症状
- 设计并实现基本操作系统功能和算法的程序
- 高级操作系统课程 – COMP 4511 Linux 中的系统和内核编程

课程大纲

- 概述（4讲） 操作系统基本概念（2讲） 系统架构（2讲）
- 进程与线程（12讲） 进程与线程（4讲） CPU调度（4讲） 同步与同步实例（2讲） 死锁（2讲）
- 内存与存储（8讲）
  - 内存管理（2讲） 虚拟内存（3讲） 辅助存储（1讲） 文件系统与实现（2讲）
- 保护（1 个讲座） 保护（1 个讲座） 安全（1 个讲座） - 可选

课程覆盖范围

- 概述
  - 第 1 章 – 操作系统的高级描述、计算机系统的基本组件（包括多处理器系统、虚拟化）
  - 第 2 章 – 操作系统服务，包括 API 和系统调用，以及常见的操作系统设计方法（单片、分层、微内核、模块化）
- 进程和线程
  - 第 3 章（进程）——进程的概念，捕获程序执行、创建和终止进程、IPC
  - 第 4 章（线程）——线程和并发执行程序的多线程进程的概念
  - 第5章（CPU调度）-CPU调度算法，包括实时调度，以及与多处理器调度和线程调度相关的问题
  - 第 6-7 章（同步）- 临界区问题、同步工具（硬件和软件）和同步示例
  - 第 8 章（死锁）——死锁特征、资源分配图、死锁预防、避免和检测算法

课程范围（续）

- 内存和存储
  - 第9章（内存）-连续内存分配、分段、分页包括分层分页
  - 第 10 章（虚拟内存）- 虚拟内存与物理内存、请求分页、页面替换算法、颠簸和帧分配第 11 章（辅助存储）- 硬盘驱动器、磁盘结构、磁盘调度算法和 RAID（磁盘阵列）结构
  - 第13-14章（文件系统）-文件访问方法、目录结构和实现、基本文件系统数据结构（磁盘上和内存中）、磁盘空间管理（包括磁盘块分配）
- 保护
  - 第17章（保护）-基本保护原则、保护环、保护域和实现（接入矩阵）
  - 第16章可选（安全）——安全威胁和攻击、安全攻击对策

第一章：简介

第一章：简介

- 操作系统的作用是什么计算机系统的组织
- 和体系结构多处理器和并行系统
-
- 操作系统的定义 虚拟化和云计算 免费开源操作系统
-

目标

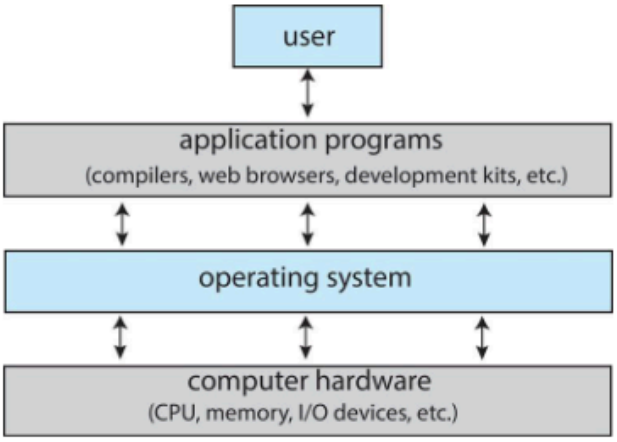
- 描述计算机系统的一般组织和中断的作用。
- 说明现代多处理器计算机系统中的组件。讨论操作系统如何在各种计算中使用
- 环境 提供免费和开源操作系统的示例

什么是操作系统?

- 用户 - 人、机器、其他计算机或设备 应用程序 - 定义如何使用系统资源来解决用户问题的方式

- 编辑器、编译器、网络浏览器、数据库、视频游戏等。 操作系统——控制和协调各种应用程序和不同用户之间计算资源的使用

- 硬件——基本计算资源，CPU、内存、I/O设备



什么是操作系统?

- 操作系统是一个程序（极其复杂），充当用户或应用程序与计算机硬件之间的中介
- Microsoft window、MacOS、iOS、Android、Linux ... 操作系统目标:
  - 执行用户程序，使解决用户问题变得更容易 使计算机系统方便使用
  - 有效地管理和使用计算机硬件
- 用户视图
  - 方便、易用、性能和安全性好 用户不关心资源利用率、效率
  -
- 系统视图
  - 操作系统作为资源分配器和控制程序

操作系统做什么

- 这取决于观点（用户或系统）和目标设备共享计算机，例如大型机或小型机
  - 操作系统需要努力让所有用户满意——性能与公平
- 各个系统（例如工作站）拥有专用资源，
  - 性能而不是公平，可以使用服务器的共享资源
- 移动设备（例如智能手机和手持设备）资源有限
  - 针对特定的用户界面，例如触摸屏、语音控制（例如 Apple 的 Siri），并针对可用性和电池寿命进行了优化
- 很少或没有用户界面的计算机或计算设备
  - 嵌入式系统 - 存在于家用设备（空调、烤面包机）、汽车、船舶、航天器中，运行实时操作系统 设计主要在无需用户干预的情况下运行 - 有些可能有数字键盘和指示灯来显示状态

操作系统定义

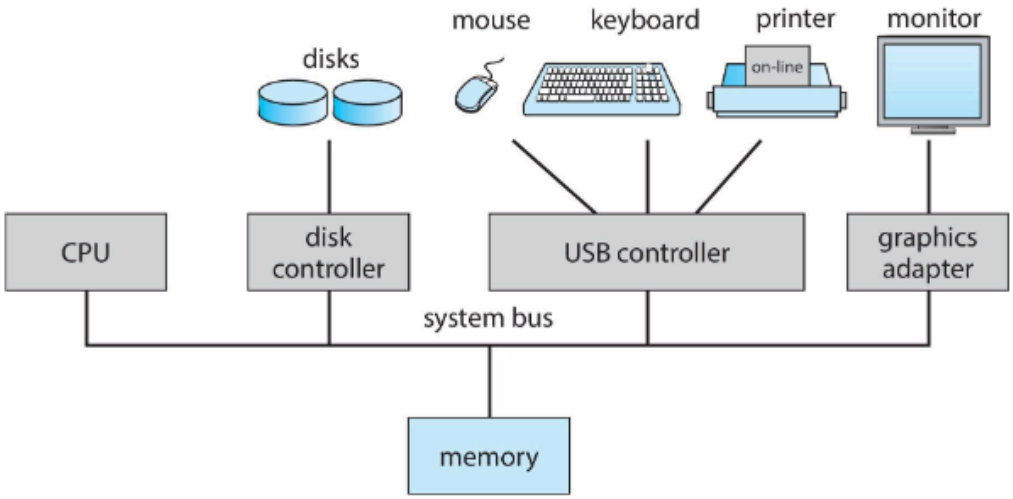
- 操作系统没有普遍接受的定义
  - “订购操作系统时供应商提供的所有东西”是一个很好的近似值，但差异很大
- 操作系统是一个资源分配器
  - 管理所有资源 - 硬件和软件 在高效和公平地使用资源的冲突请求之间做出决定
- OS是一个控制程序
  - 控制程序的执行，防止错误和计算机的不当使用
- 简而言之，操作系统管理和控制硬件并帮助程序在计算机上运行。

操作系统定义

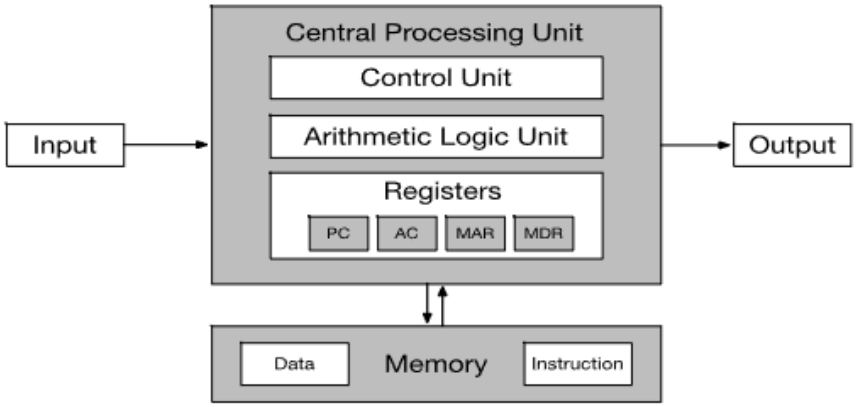
- 核心
  - “在计算机上始终运行的一个程序”基本功能 - 在本入门课程中讨论
  -
- 中间件
  - 一组软件框架，为应用程序开发人员提供数据库、多媒体、图形等附加服务，流行于移动操作系统 - Apple 的 iOS 和 Google 的 Android
  -
- 其他一切
  - 系统程序（随操作系统一起提供，但不是内核的一部分），例如文字处理器、浏览器、编译器
  - 与操作系统无关的应用程序——apps
- 操作系统包括始终运行的内核、简化应用程序开发并提供附加功能的中间件框架，以及在系统运行时帮助管理系统的系统程序

计算机系统组织

- 电脑系统操作
  - 一个或多个 CPU 核心、通过公共总线连接的设备控制器，提供对共享内存的访问
  - CPU和设备的并发执行——通过共享总线竞争内存周期

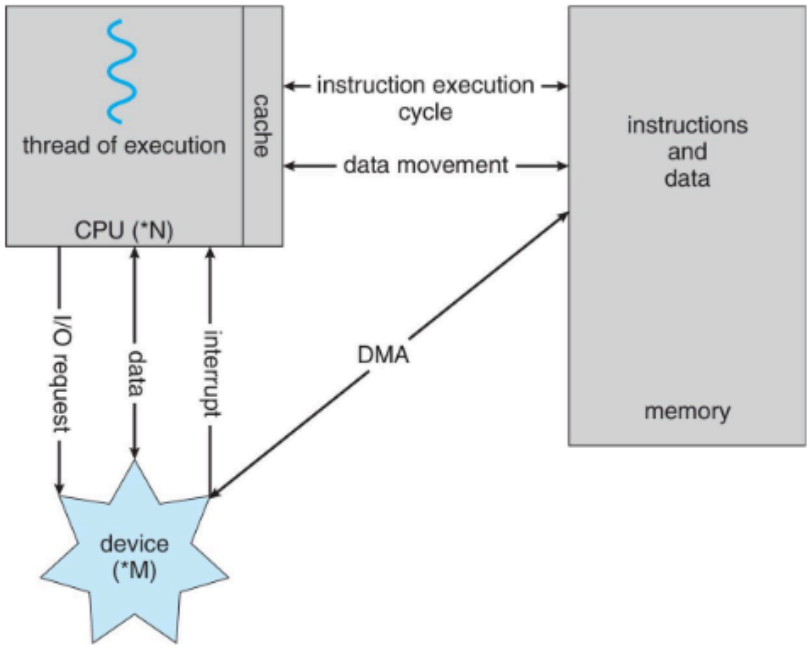


冯诺依曼架构



- 包含算术逻辑单元 (ALU) 和处理器寄存器的中央处理单元 – 程序计数器 (PC)、累加器 (AC)、内存地址寄存器 (MAR)、内存数据寄存器 (MDR) 包含指令寄存器 (MDR) 的控制单元IR）和程序计数器（PC）
- 内存存储数据和指令 - 以及缓存 外部大容量存储 - 辅助存储（图中未显示） 输入和输出机制
-

现代计算机的工作原理



冯诺依曼架构

执行指令的步骤：

- 取指令 解码指令
- 取数据 执行指令
- 如有则回写

指令获取/解码/执行

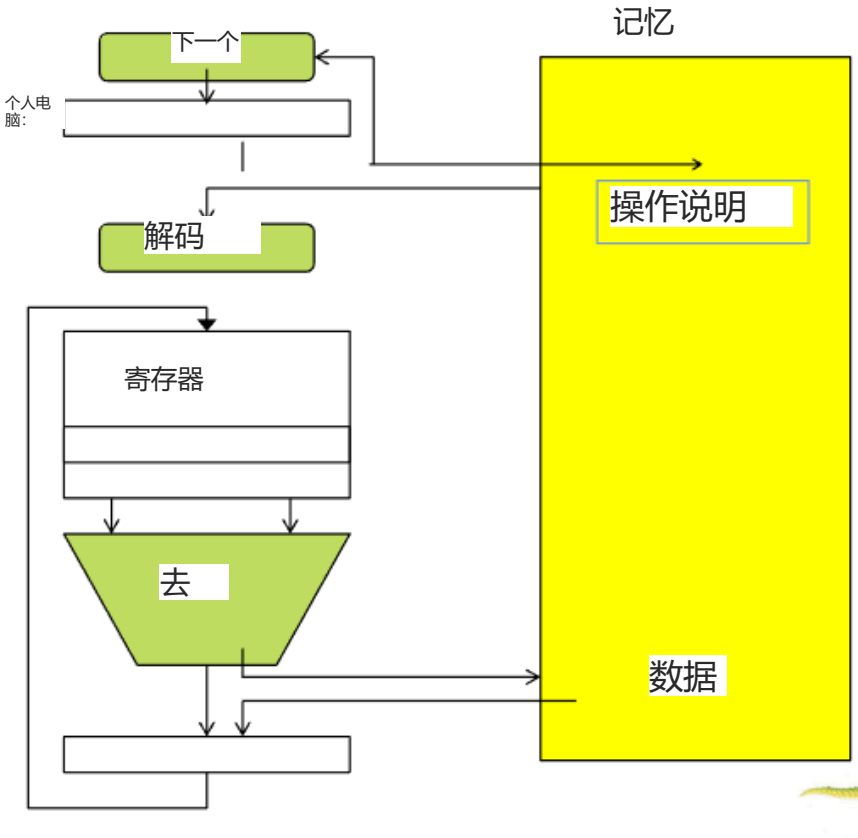
指令周期

处理器

取指令

解码

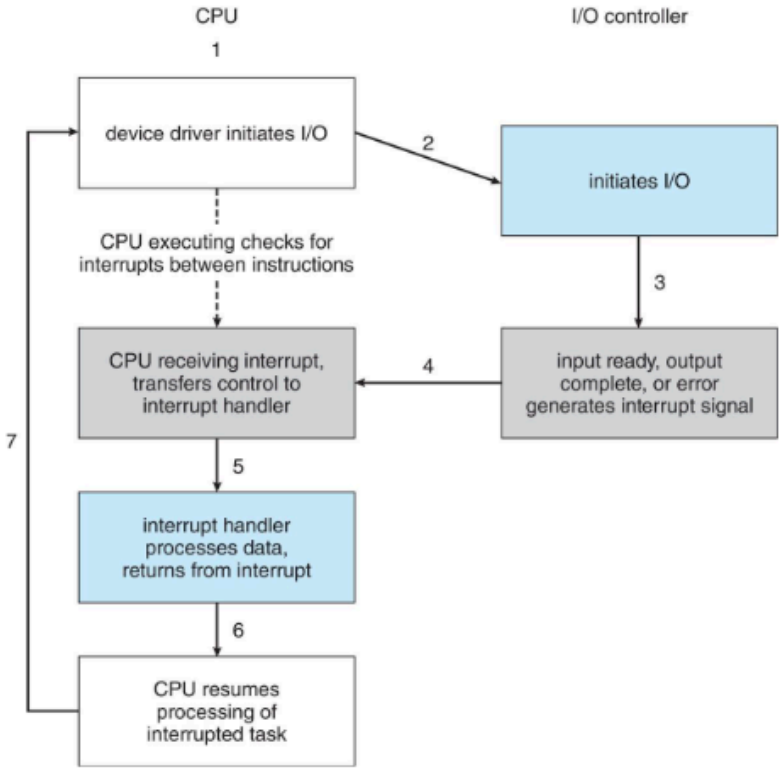
执行



计算机系统操作 – I/O

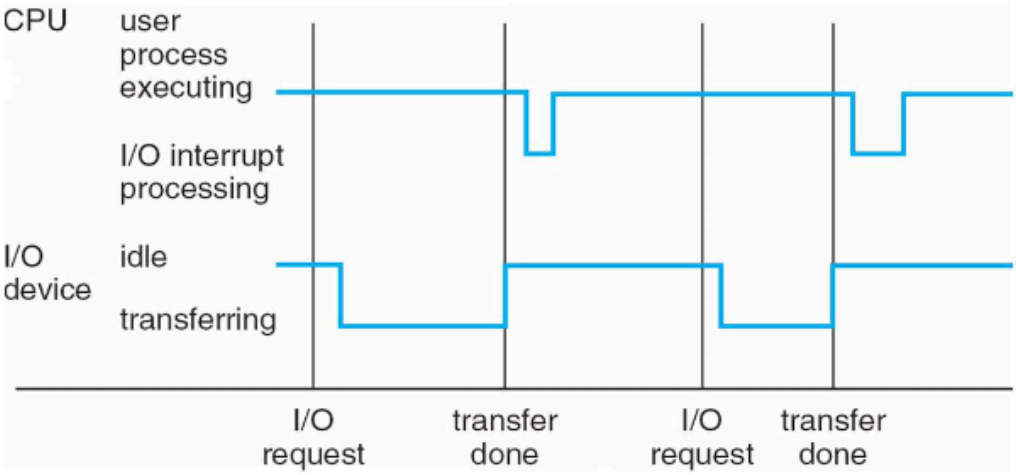
- I/O 设备和 CPU 并发异步执行 每个设备控制器负责特定设备
- 每个设备控制器都有一个本地缓冲区
-
- 设备控制器负责在其控制的外围设备与其本地缓冲存储器之间移动数据
  - I/O操作是从设备到控制器的本地缓冲区
- CPU 将数据从主内存移至本地缓冲区或从本地缓冲区移出，通常用于键盘和鼠标等慢速设备 DMA 控制器用于将数据移至磁盘等快速设备 设备控制器通过引发中断——需要CPU关注
-
  - 对于输入设备，这意味着数据在本地缓冲区中可用 对于输出设备，它通知 CPU I/O 操作已完成
  -

中断驱动的 I/O 周期


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

中断时间线

- CPU 和设备同时执行 I/O 设备可能会通过向 CPU 发送信号来触发中断 CPU 处理该中断，然后返回到被中断的指令



中断的常用功能

- 中断在现代操作系统中广泛用于处理异步事件 - 设备控制器和硬件故障 中断将控制转移到中断服务例程或中断处理程序 - 内
- 核代码的一部分，操作系统运行该代码来处理特定中断

- 中断机制还实现了中断优先级系统，使得高优先级中断可以抢占低优先级中断的执行

- 陷阱或异常是由错误（例如算术错误）或用户请求（例如请求操作系统服务的系统调用 - 待讨论）引起的软件生成的中断

- 所有现代操作系统都是中断驱动的 在现代计算机系统中，每秒会发
- 生数百个中断 - 因为 CPU 运行速度极快，不到一纳秒

存储定义和符号审查

计算机存储的基本单位是位。一位可以包含两个值之一：0 和 1。计算机中的所有其他存储都基于位的集合。如果有足够的位，计算机可以表示的东西之多令人惊讶：数字、字母、图像、电影、声音、文档和程序等等。一个字节是 8 位，在大多数计算机上，它是最小的方便存储块。例如，大多数计算机没有移动一位的指令，但有移动一个字节的指令。一个不太常见的术语是字，它是给定计算机体系结构的本机数据单位。一个字由一个或多个字节组成。例如，具有 64 位寄存器和 64 位内存寻址的计算机通常具有 64 位（8 字节）字。计算机以其本机字大小而不是一次一个字节执行许多操作。

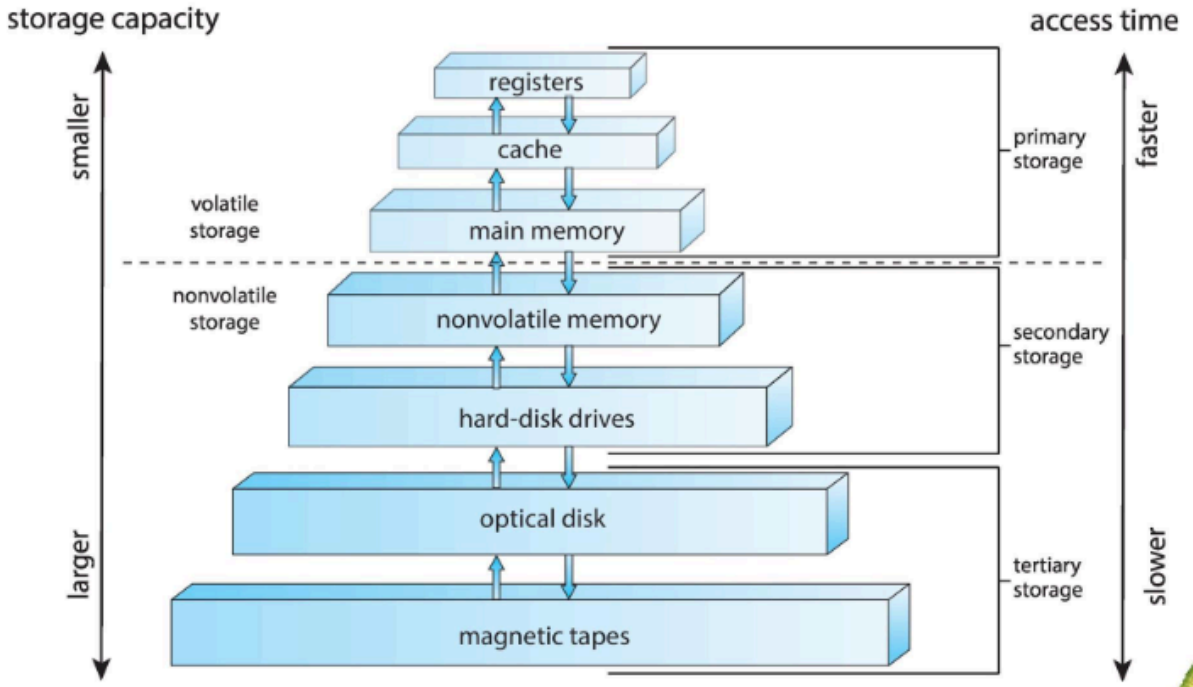
计算机存储以及大多数计算机吞吐量通常以字节和字节集合来测量和操作。

千字节 (KB) 为 1,024 字节，兆字节 (MB) 为 1,0242 字节，千兆字节 (GB) 为 1,0243 字节，太字节 (TB) 为 1,0244 字节，拍字节 (PB) 为 1 ,0245 字节

计算机制造商经常对这些数字进行四舍五入，并称 1 MB 为 100 万字节，1 GB 为 10 亿字节。网络测量是此一般规则的一个例外；它们以位为单位（因为网络一次移动一点数据）。

存储层次结构

- 存储系统按层次结构组织，随速度、单位成本、容量（大小）和易失性（非易失性磁盘与易失性内存）的变化而变化



记忆

- 主存储器 – CPU 可以直接访问的唯一大型存储介质 易失性，通常是动态随机存取存储器 (DRAM) 形式的随机存取存储器
  - 基本操作将指令加载和存储到特定的内存地址，这是可字节寻址的——每个地址引用内存中的一个字节
- 计算机也使用其他形式的内存。例如，计算机加电时运行的第一个程序是引导程序，该程序存储在电可擦除可编程只读存储器（EEPROM）中

第二存储

- 辅助存储——主存储器的扩展，提供大的非易失性存储容量，可以永久保存大量数据。
- 最常见的辅助存储设备是硬盘驱动器 (HDD) 和非易失性存储器 (NVM) 设备，它们为程序和数据提供存储。
- 二级存储一般有两种类型
  - 机械，例如 HDD、光盘、全息存储和磁带
  - 电气方面，例如闪存、SSD、FRAM、NRAM。电存储通常称为NVM
- 机械存储通常比电存储更大且每字节更便宜。相反，电存储通常比机械存储成本更高、更小、更可靠且更快。

缓存

- 重要原则 - 在计算机中的多个级别上执行
  - 内存、地址转换、文件块、文件名（常用）、文件目录、网络路由等的缓存。
- 基本思想：暂时从较慢的存储复制到较快的存储的信息子集
- 使频繁使用的情况更快，使不那么频繁的情况不那么占主导地位访问首先检查以确定信息是否在缓存内
  - Hit：如果是，则直接从缓存使用信息（快速） Miss：如果不是，则将数据从较慢的存储复制到缓存并在那里使用
- 缓存通常比被缓存的存储（例如内存）小得多
  - 缓存管理：缓存大小和替换策略主要标准——缓存命中率；在缓存中找到的内容百分比
- 重要衡量指标 平均访问时间 = (命中率 x 命中时间) + (未命中率 x 未命中时间)

为什么缓存有效？ - 地点

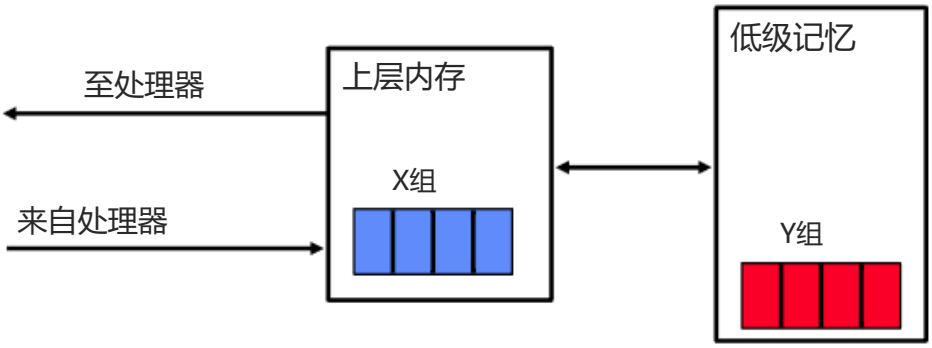
- 时间局部性（时间局部性）0 地址空间 2n - 1

• 时间局部性（时间局部性）： – 使最近访问的数据项更靠近处理器

• 空间局部性（空间局部性）： – 将连续的块移动到上层

序）

则缓存将永



各类存储的特点

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

存储层次结构级别之间的移动可以是显式的或隐式的

时间尺度范围

杰夫·迪恩： “每个人都应该知道的数字”

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

输入/输出子系统

- 操作系统需要适应各种各样的设备，每种设备都有不同的功能、控制位定义以及与主机操作系统交互的协议，使 I/O 设备能够以标准、统一的方式进行处理——这涉及抽象、封装和软件分层，就像任何复杂的软件工程设计一样
-

- I/O 系统调用将设备行为封装在几个通用类中，每个类都通过一组标准化的函数（接口）进行访问 OS 的一个目的是向用户隐藏硬件设备的特殊性 I/O 子系统负责
-
-

  - I/O 的内存管理，包括缓冲（在传输数据时临时存储数据）、缓存（将部分数据存储在更快的存储中，以便

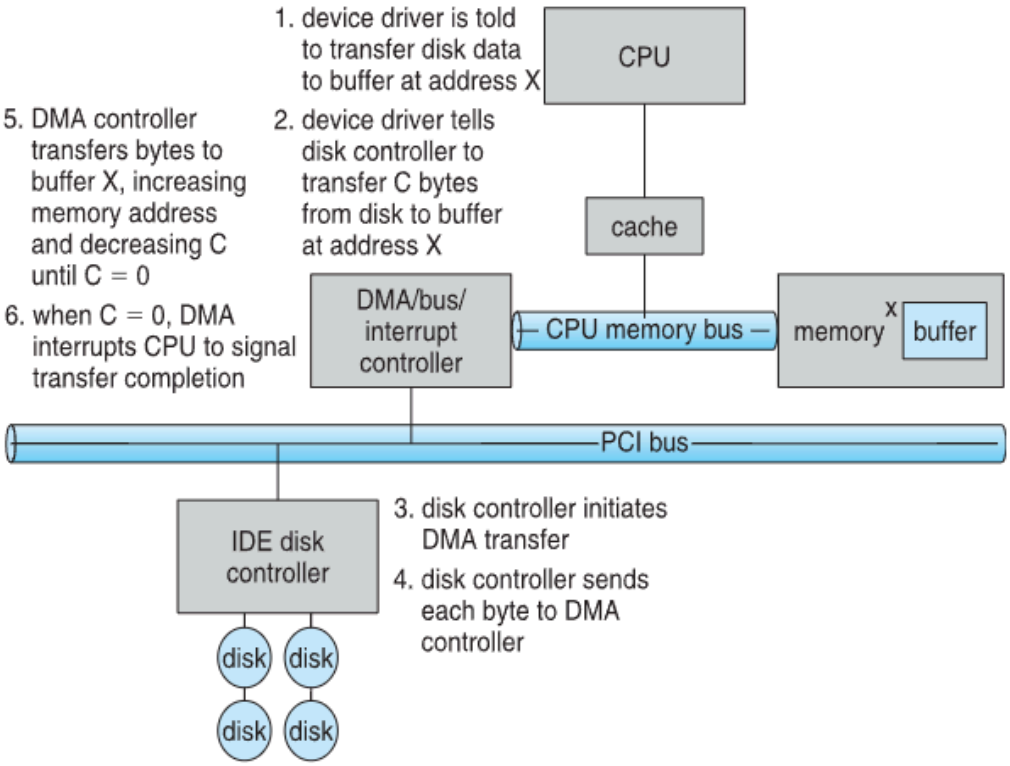
    性能）、假脱机（一项作业的输出与其他作业的输入重叠，通常用于打印机） 通用设备驱动程序接口
  -
  - 特定硬件设备的驱动程序

直接内存访问

- 编程 I/O - CPU 运行特殊的 I/O 指令，在内存和慢速设备（例如键盘和鼠标）之间一次移动一个字节 为了避免编程 I/O，对于快速设备和大量数据传输，它使用直接内存访问或 DMA 控制器 - 绕过 CPU 在 I/O 设备和内存之间直接传输数据 - CPU 或操作系统初始化 DMA 控制器，DMA 控制器负责在设备和内存之间移动数据，无需 CPU 参与。

- 这使 CPU 免于缓慢的数据移动（I/O 操作） OS 将 DMA 命令块写入内存
  - 源地址和目标地址 读或写模式
  - 要传输的字节数 将命令块的位置写入 DMA 控制器 DMA 控制器的总线主控 – 从 CPU 获取总线 完成后，向 CPU 发送中断以指示完成

执行 DMA 传输的六步过程



单处理器系统

- 过去，大多数计算机系统使用单个处理器，其中包含一个具有单个处理核心的 CPU
  - 核心执行指令和寄存器以在本地存储数据。处理核心或CPU核心能够执行通用指令集
- 此类系统还有其他专用处理器 - 设备特定处理器，例如磁盘和图形控制器 (GPU)。
  - 它们运行有限的指令集，通常不执行用户进程的指令

多处理器系统

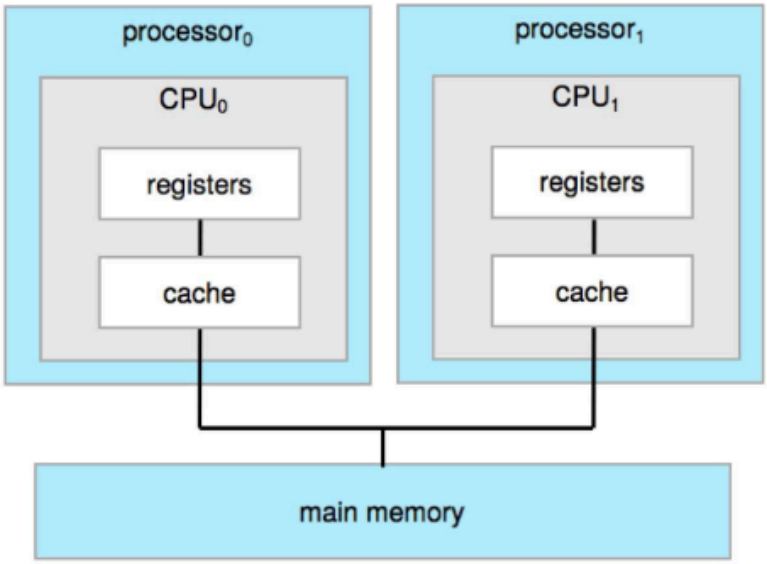
- 在现代计算机上，从移动设备到服务器，多处理器系统现在主导着计算领域
  - 传统上，此类系统有两个（或更多）处理器，每个处理器都有一个单核 CPU

由于开销，例如共享资源（总线或内存）的争用，N 个处理器的加速比小于 N。多处理器系统的使用和重要性不断增长，优点是

-
  - 更高的吞吐量 - 更强的计算能力 规模经济 - 共享其他设备，例如 I/O 设备 更高的可靠性 - 优雅降级或容错
  -
  -
- 两种类型的多处理器系统
  - 非对称多处理 – 通常采用主从方式，主处理器将特定任务分配给从处理器，主处理器处理 I/O 对称多处理 – 每个处理器执行所有任务，包括操作系统功能和用户进程
  -

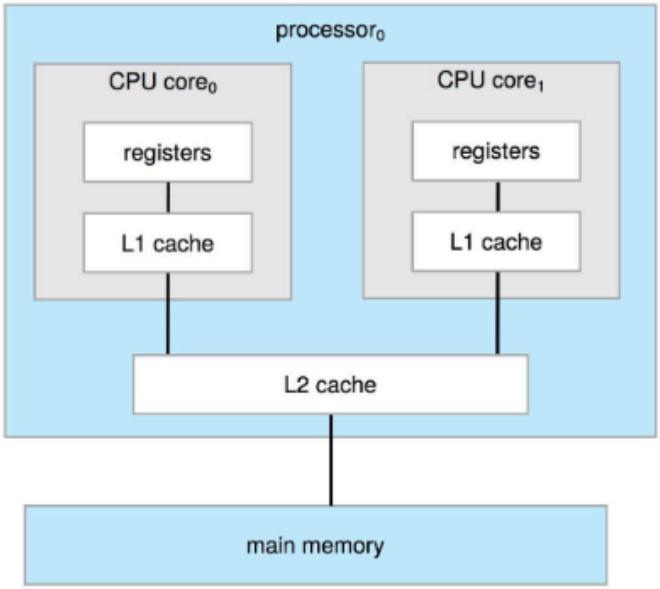
对称多处理器系统

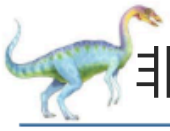
- 对称多处理或 SMP – 每个 CPU 处理器都有自己的一组寄存器，以及私有或本地缓存。然而，所有处理器通过系统总线共享物理内存。



多核设计

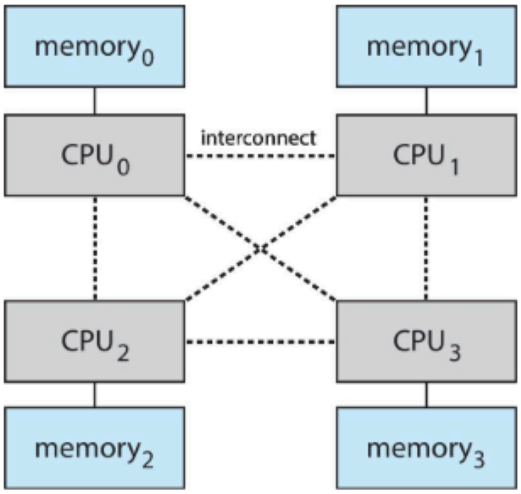
- 多核、多个计算核心驻留在单个物理芯片上 片上通信比芯片间通信更快 功耗显着降低 - 对于移动设备和笔记本电脑非常重要





非统一内存访问 (NUMA)

- 由于系统总线的争用，向多处理器系统添加更多 CPU 可能无法扩展，这可能会成为瓶颈
- 另一种方法是为每个 CPU（或 CPU 组）提供自己的本地内存，可通过小型、快速的本地总线进行访问。
- CPU 通过共享系统互连进行连接，并且所有 CPU 共享一个物理内存地址空间。这种方法称为非均匀内存访问或 NUMA
- NUMA 系统的潜在缺点是，当 CPU 必须通过系统互连访问远程内存时，延迟会增加 - 调度和内存管理的影响



计算机系统组件

- CPU - 执行指令的硬件 处理器 - 包含一个或多个 CPU 的物理芯片 核心 - CPU 的基本计算单元或执行指令和本地存储数据的寄存器的组件 多核 - 包括单个物理处理器上的多个计算核心芯片
- 多处理器系统——包括多个处理器

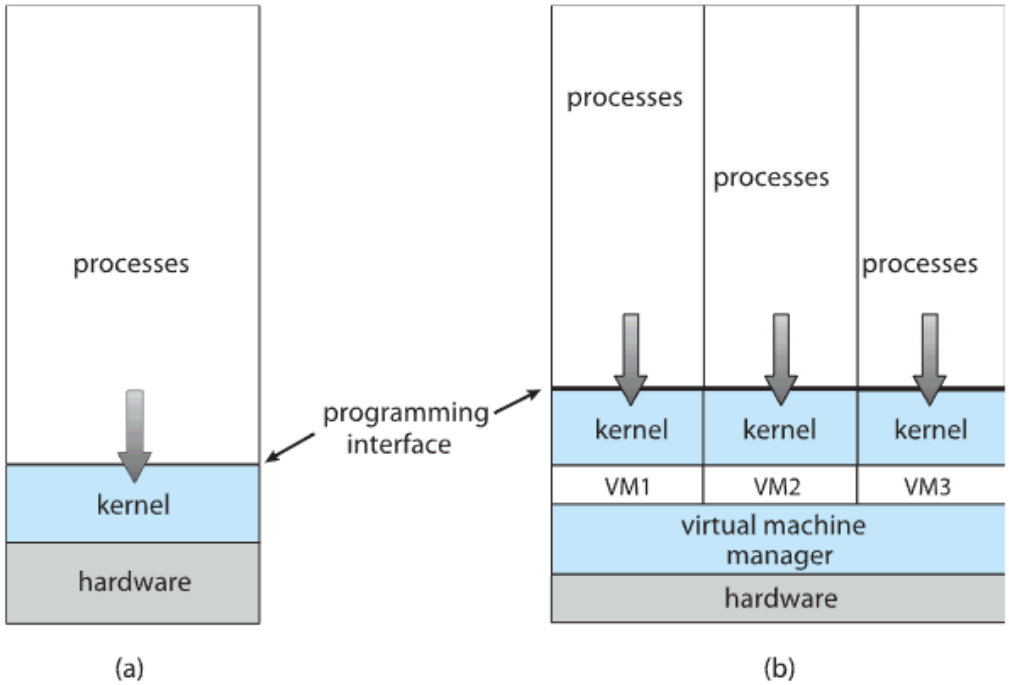
操作系统结构

- 所有现代操作系统都有两个共同特征：为了提高效率，需要进行多道程序设计（批处理系统）
  - 在过去，操作系统一次将一个程序加载到内存中执行。单个程序不能总是让 CPU 或 I/O 设备保持忙碌，因为它们变得越来越快——所有现代计算机系统都是多道程序的。多道程序以一种希望的方式组织作业。 CPU总是有一个要执行的在大型机中，作业被远程提交并排队，作业通过作业调度选择并运行——加载到内存中（稍后讨论）
  -
  -
- 分时（多任务）是多道程序设计的逻辑扩展，其中CPU在作业之间“频繁”切换，用户可以在每个作业运行时与其进行交互，从而实现交互式计算
  - 响应时间应小于 1 秒 每个用户至少有一个程序在内存中执行 □ 进程
  - 如果有多个作业准备同时运行 □ CPU 调度 如果内存无法容纳进程，则交换技术会在执行过程中将它们移入和移出内存
  -
  - 虚拟内存允许不完全在内存中执行进程

虚拟化

- 虚拟化将单个计算机的硬件抽象为多个不同的执行环境 - 造成每个用户或程序都在自己的“私人计算机”上运行的错觉
  - 它创建一个虚拟系统 - 虚拟机或 VM，操作系统和应用程序可以在其上运行
  - 它还允许操作系统作为其他操作系统中的应用程序运行——这是一个巨大且不断发展的行业
- 几个组件
  - 主机 – 底层硬件系统 虚拟机管理器 (VMM) 或管理程序 – 通过提供与主机相同的接口来创建和运行虚拟机 来宾 – 提供主机虚拟副本的进程，通常是操作系统 – 来宾操作系统
  -
- 这使得单个物理机可以同时运行多个操作系统，每个操作系统都在自己的虚拟机中

虚拟化 – 系统模型

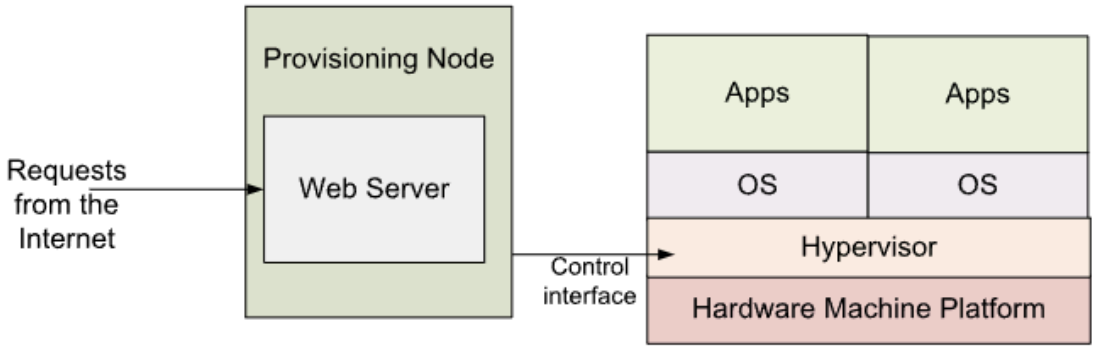

processes
kernel
hardware
programming interface
processes
processes
processes
kernel
kernel
kernel
VM1
VM2
VM3
virtual machine manager
hardware
(a)
(b)

虚拟化——一点历史

- 虚拟化 – 为 CPU 本机编译的操作系统，运行来宾操作系统
  - 虚拟化最初是在 IBM 大型机 (1972) 中设计的，允许多个用户在为单个用户设计的系统中同时运行任务或共享面向批处理的系统
  - VMware 在 Intel x86 CPU 上运行一个或多个 Windows 来宾副本，每个副本都运行自己的应用程序
  - 虚拟机管理器或 VMM 为程序提供了一个与原始计算机（界面）基本相同的环境。在此类环境中运行的程序仅显示出较小的差异。
  - 性能下降 – 通过更多的软件层 VMM 完全控制系统资源
  -
- 20 世纪 90 年代末 Intel CPU 足够快 - 通用 PC 上的虚拟化
  - Xen 和 VMware 创建的技术至今仍在使用 虚拟化已扩展到许多操作系统、CPU、VMM
  -

云计算和虚拟化

- 通过网络将计算、存储和应用程序作为服务提供 虚拟化的逻辑扩展，因为它使用虚拟化作为其功能的基础。
  - Amazon EC2 拥有数百万台服务器、数千万个虚拟机、通过 Internet 提供的 PB 存储空间，按使用量付费



云计算类型

- 云的种类很多
  - 公共云 – 任何愿意付费的人都可以通过互联网使用 私有云 – 由公司运行供公司自己使用 混合云 – 包括公共云和私有云组件 软件即服务 (SaaS) – 通过互联网提供一个或多个应用程序（即文字处理器）
  - 平台即服务 (PaaS) – 可供应用程序通过互联网使用的软件堆栈（即数据库服务器）
  - 基础设施即服务 (IaaS) – 通过互联网提供的服务器或存储（即可用于备份的存储）
  - 越来越多地提供其他服务，例如 MaaS 或机器学习即服务

免费开源操作系统

- 操作系统以源代码格式提供，而不仅仅是二进制、闭源和专有
  - Microsoft Windows 是闭源方法的一个著名例子。
- 由自由软件基金会 (FSF) 发起，该基金会拥有“copyleft”GNU 公共许可证 (GPL)
  - 自由软件和开源软件是两种不同的想法 http://gnu.org/philosophy/open-source-misses-the-point.html/ 自由软件不仅提供源代码，而且还被许可不允许任何成本使用、重新分配和修改。开源软件不一定提供此类许可
- 流行的例子包括 GNU/Linux、FreeBSD UNIX（包括 Mac OS X 的核心 - Darwin）和 Solaris
- 开源代码可以说更安全，允许更多程序员做出贡献，并且无疑是更好的学习工具

第一章结束