CS162
Operating Systems and
Systems Programming
Lecture 1

Overview

August 29th, 2011
Anthony D. Joseph and Ion Stoica
http://inst.eecs.berkeley.edu/~cs162

Who Are We?

- Professor Anthony D. Joseph
  - 465 Soda Hall (RAD Lab)
  - Web: http://www.cs.berkeley.edu/~adj/
  - Office hours (tentative): MTu 2-3pm in 465D Soda

- Research areas:
  - Current: Cloud computing (Mesos), Secure Machine Learning (SecML), DETER security testbed
  - Other: P2P (Tapestry), Mobile computing, Wireless/ Cellular networking

---

Who Are We?

- Ion Stoica
  - E-mail: istoica@cs.berkeley.edu
  - Web: http://www.cs.berkeley.edu/~istoica/
  - Office hours (tentative): M 3-4PM, W11-12 in 465D Soda

- Research focus
  - Cloud computing (Mesos, Spark)
  - Network architectures (i3, Declarative Networks, …)
  - P2P (Chord, OpenDHT)
  - Tracing and debugging in distributed systems (ODR, Liblog, Friday)

TAs

- Angela Juang
  - E-mail: cs162-ta@cory.eecs.berkeley.edu
  - Sections: Tu 10-11am and 11-12pm
  - Office hours: TBA (place: TBA)

- Steve Hanna
  - E-mail: cs162-tb@cory.eecs.berkeley.edu
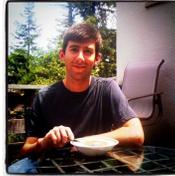 - Sections: Tu 11-12pm
  - Office hours: TBA (place: TBA)

TAs (cont'd)

- Karan Malik
  - E-mail: cs162-tc@cory.eecs.berkeley.edu
  - Sections: Tu 3-4pm and 6-7pm
  - Office hours: TBA

- Patrik Sundberg
  - E-mail: cs162-td@cory.eecs.berkeley.edu
  - Sections: Tu 9-10 and 10-11am
  - Office hours: TBA

TAs (cont'd)

- Andrew Wang
  - E-mail: cs162-te@cory.eecs.berkeley.edu
  - Sections: Tu 4-5pm and 5-6pm
  - Office hours: TBA

Goals of Today Lecture

- What are we going to learn and why?

- What is an operating system?

- How does this class operate?

- Interactivity is important! Please ask questions!

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne. Slides courtesy of Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.
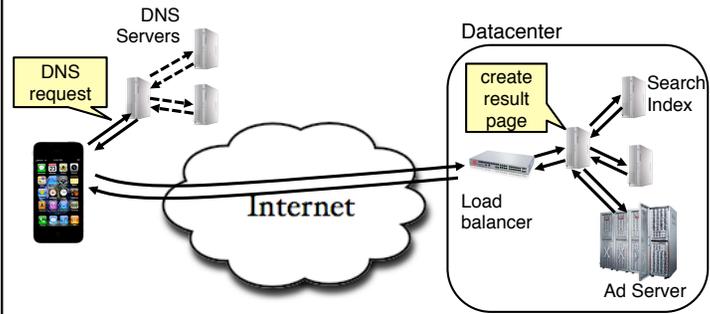
Goal of This Course

- Learn how "systems" work

- Main challenges in building systems

- Principles of system design, i.e., how to address to challenges

- Learn how to apply these principles to system design

Example: What's in a Search Query?

- Complex interaction of multiple components in multiple administrative domains

Computing Devices Everywhere

Challenges

- Enormous scale, heterogeneity, and dynamic range:
  - CPU: sensor motes → GPUs
    - » Cores: one → 100s   [2-orders of magnitude variation]
    - » Clusters: few machines → 10,000s machines [4 orders of mag.]
  - Network: Inter-core networks → Internet
    - » Latency: nanosecs → secs (satellite) [9 orders of mag.]
    - » Bandwidth: Kbps → Gbps   [6 orders of mag.]
    - » …
  - Storage: caches → disks
    - » Size: MB → TB   [6 orders of mag.]
    - » Access time: few nanosecs → millisecs [6 orders of mag.]
- Complexity
  - Complex interaction between system components
  - Unexpected failure scenarios, e.g., randomly flipping a memory bit

Technology Trends: Moore's Law
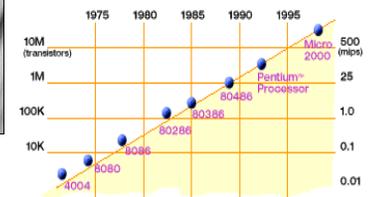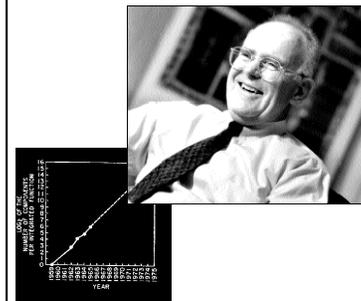
2X transistors/Chip Every 1.5 years
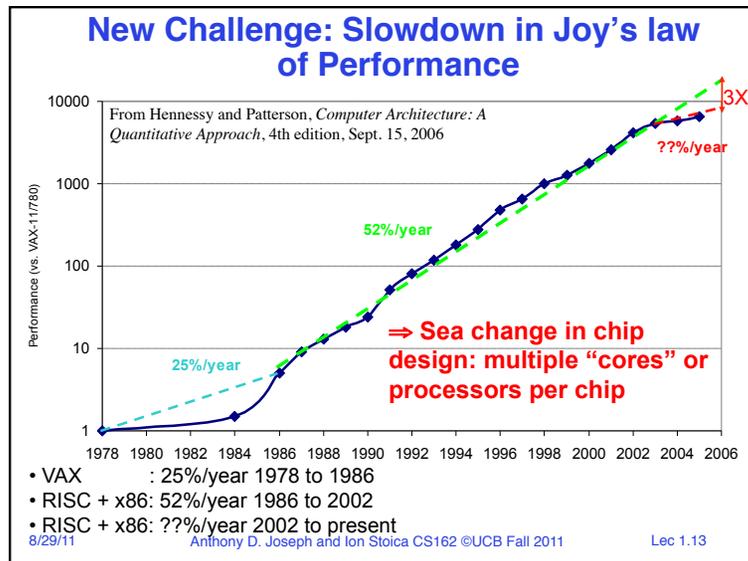Called "Moore's Law"

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Microprocessors have become smaller, denser, and more powerful.

Page 3

New Challenge: Slowdown in Joy's law of Performance

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

3X

??%/year

52%/year

25%/year

⇒ **Sea change in chip design: multiple "cores" or processors per chip**

- VAX   : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
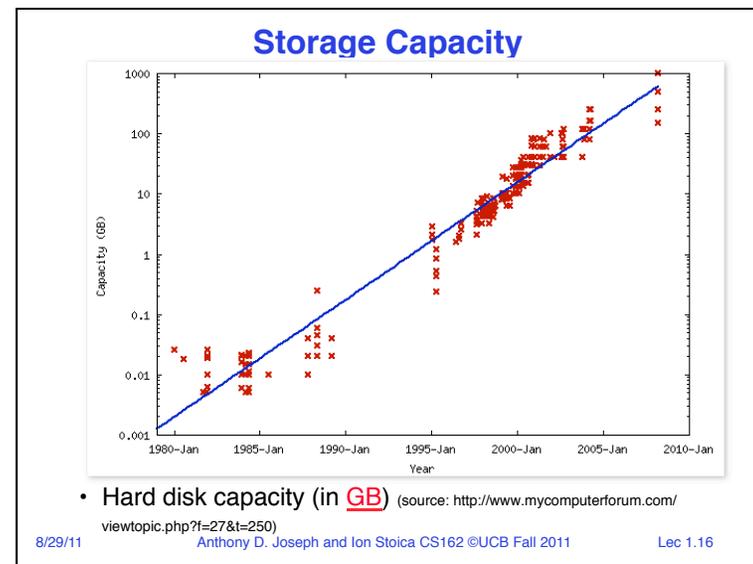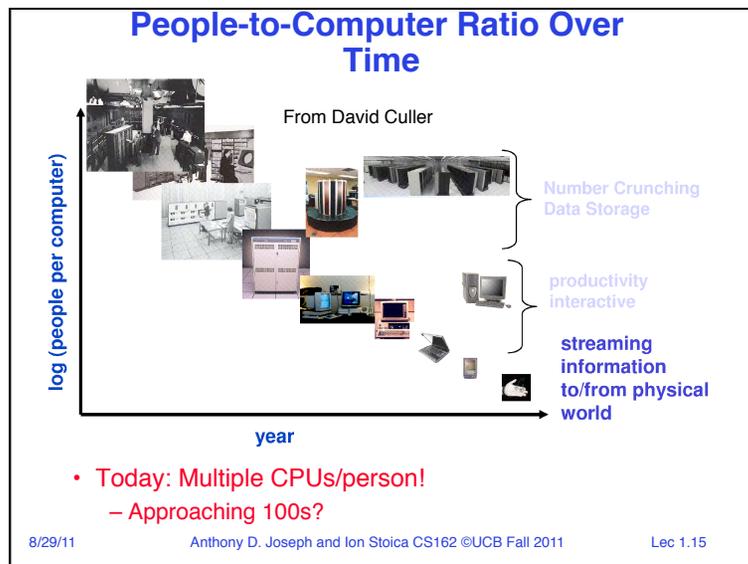- RISC + x86: ??%/year 2002 to present

ManyCore Chips: The future is here

- Intel 80-core multicore chip (Feb 2007)
  – 80 simple cores
  – Two FP-engines / core
  – Mesh-like network
  – 100 million transistors
  – 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  – 24 "tiles" with two cores/tile
  – 24-router mesh network
  – 4 DDR3 memory controllers
  – Hardware support for message-passing

- "ManyCore" refers to many processors/chip
  – 64? 128? Hard to say exact boundary
- How to program these?
  – Use 2 CPUs for video/audio
  – Use 1 for word processor, 1 for browser
  – 76 for virus checking???
- Parallelism must be exploited at all levels

People-to-Computer Ratio Over Time

From David Culler

log (people per computer)

Number Crunching Data Storage

productivity interactive

streaming information to/from physical world

year

- Today: Multiple CPUs/person!
  – Approaching 100s?

Storage Capacity

- Hard disk capacity (in GB) (source: http://www.mycomputerforum.com/ viewtopic.php?f=27&t=250)

Page 4

Breaking News: New IBM Storage Array

- IBM's Almaden research lab
- 120 Petabyte array (120 million Gigabytes)
  - 200,000 conventional hard disk drives

- Perspective
  - Roughly one trillion files
  - Today's largest arrays: 15PB
  - 24 billion typical 5MB MP3 files
  - 60 copies of the 150 billion pages in the Internet Archive's WayBack Machine
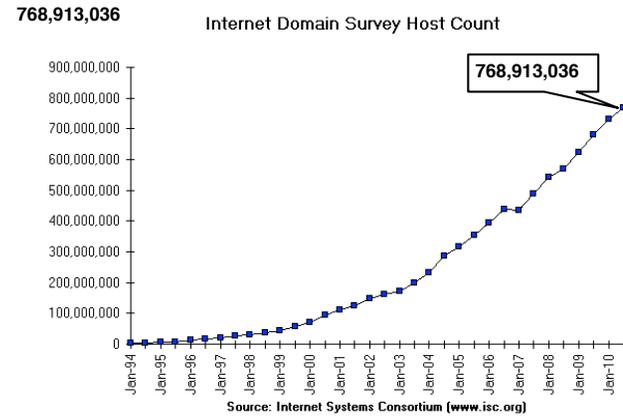 - GPFS handles many drive failures per day

Source: MIT Technology Review (http://www.technologyreview.com/computing/38440/)

Internet Scale: .75 Billion Hosts

768,913,036

Internet Domain Survey Host Count

768,913,036

Source: Internet Systems Consortium (www.isc.org)

Internet Scale: Two Billion Users!

WORLD INTERNET USAGE AND POPULATION STATISTICS						
World Regions	Population (2010 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2010	Users % of Table
Africa	1,013,779,050	4,514,400	110,931,700	10.9 %	2,357.3 %	5.6 %
Asia	3,834,792,852	114,304,000	825,094,396	21.5 %	621.8 %	42.0 %
Europe	813,319,511	105,096,093	475,069,448	58.4 %	352.0 %	24.2 %
Middle East	212,336,924	3,284,800	63,240,946	29.8 %	1,825.3 %	3.2 %
North America	344,124,450	108,096,800	266,224,500	77.4 %	146.3 %	13.5 %
Latin America/Caribbean	592,556,972	18,068,919	204,689,836	34.5 %	1,032.8 %	10.4 %
Oceania / Australia	34,700,201	7,620,480	21,263,990	61.3 %	179.0 %	1.1 %
WORLD TOTAL	6,845,609,960	360,985,492	1,966,514,816	28.7 %	444.8 %	100.0 %

Societal Scale Information Systems

- The world is a large parallel system
  - Microprocessors in everything
  - Vast infrastructure behind them

Massive Cluster

Clusters

Internet Connectivity

Scalable, Reliable, Secure Services
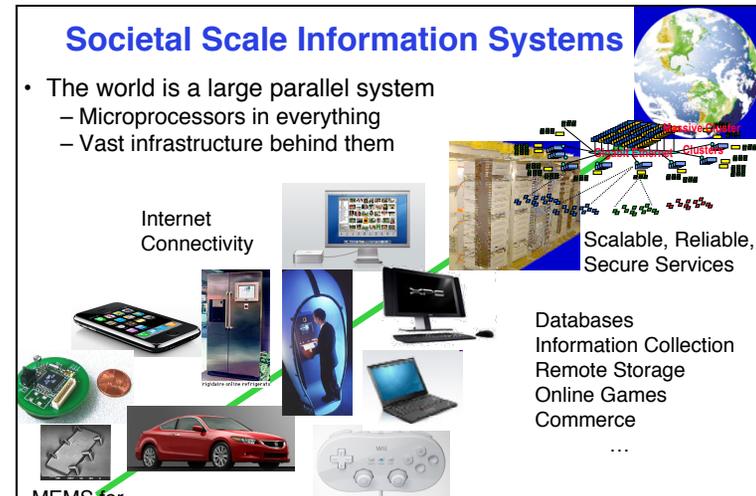
Databases
Information Collection
Remote Storage
Online Games
Commerce
…

MEMS for Sensor Nets

Page 5

Class Schedule & Info

- Class Time: MW 4-5:30pm, 4 Leconte
  - Please come to class; best part of class is interaction!
  - Also: 5% of grade is from class participation (section and class)

- Sections
  - Important information is in the sections
  - The sections assigned to you by Telebears are temporary!
  - Every member of a project group must be in same section

- Website: http://www-inst.eecs.berkeley.edu/~cs162/
- Newsgroup: http://www.piazza.com/
- Time-delayed Webcast (audio podcast with slides)

Lecture Goal

Interactive!!!

Syllabus

- Different emphasis from CS162 in previous years; beginning this year we have shifted to emphasize
  - End-to-end system design, rather than OS only
  - More networking, database, and security concepts
  - New projects to reflect this emphasis

- Long term plan: make CS 162 a gateway course for
  - Database class (CS 186)
  - Networking class (EE 122)
  - Security class (CS 161)
  - Software engineering class (CS 169)
  - New operating systems class (cs16x)

Syllabus (cont'd)

- 11 lectures on OS

- 4 lectures on Networking

- 3 lectures on Databases

- 2 lectures on Security

- 1 lecture on Software Engineering

- 3 capstone lectures on "putting everything together" (client-server, cloud computing, P2P)

Textbook

- Textbook: Operating Systems Concepts, 8th Edition Silbershatz, Galvin, Gagne

- Online supplements
  - See "Information" link on course website
  - Includes Appendices, sample problems, etc

- Networking, Databases, Software Eng
  - Limited to lecture notes

Grading

- Rough Grade Breakdown
  - Midterm: 20%
  - Final: 25%
  - Four Projects: 50% (i.e. 12.5% each)
  - Participation: 5%

- Four Projects: build a fully-functional game of Go hosted in the cloud (server and machine players)
  - Phase I: Single-machine, single process Go game
  - Phase II: Client-server communication
  - Phase III: Go server on Amazon EC2 w/ game DB
  - Phase IV: Distributed Go client on Amazon EC2
  - End of semester contest

- Late Policy:
  - No slip days!
  - 10% off per day after deadline

Computing Facilities

- Every student who is enrolled should get an account form at the end of lecture
  - Gives you an account of form cs162-xx@cory
  - This account is required
    » Most of your debugging can be done on other EECS accounts, however…
    » All of the final runs must be done on your cs162-xx account and must run on the x86 Solaris machines

- Make sure to log in into your new account this week

Group Project Simulates Industrial Environment

- Project teams have 4 or 5 members in same discussion section
  - Must work in groups in "the real world"
- Communicate with colleagues (team members)
  - Communication problems are natural
  - What have you done?
  - What answers you need from others?
  - You must document your work!!!
- Communicate with supervisor (TAs)
  - How is the team's plan?
  - Short progress reports are required:
    » What is the team's game plan?
    » What is each member's responsibility?

Project Signup

- Project Signup: Watch "Group/Section Signup" Link
  - 4-5 members to a group
    » Everyone in group must be able to *actually* attend same section
    » The sections assigned to you by Telebears are temporary!
  - Only submit once per group! Due Friday (9/2) by 11:59pm
    » Everyone in group must have logged into their cs162-xx accounts once before you register the group
    » Make sure that you select at least 2 potential sections
- New section assignments emailed next Monday/Tuesday
  - Attend new sections next week

Section	Time	Location	TA
107	Tu 9:00A-10:00A	320 Soda	Patrik
101, 108	Tu 10:00A-11:00P	87 Evans, 320 Soda	Patrik, Angela
102, 109	Tu 11:00A-12:00P	87 Evans, 320 Soda	Steve, Angela
103	Tu 3:00P-4:00P	85 Evans	Karan
104	Tu 4:00P-5:00P	85 Evans	Andrew
105	Tu 5:00P-6:00P	85 Evans	Andrew
106	Tu 6:00P-7:00P	259 Dwinelle (TEMP)	Karan

5min Break

Complexity

- Applications consisting of…
  - … a variety of software modules that …
  - … run on a variety of devices (machines) that
    » … implement different hardware architectures
    » … run competing applications
    » … fail in unexpected ways
    » … can be under a variety of attacks

- Not feasible to test software for all possible environments and combinations of components and devices
  - The question is not whether there are bugs but how serious are the bugs!

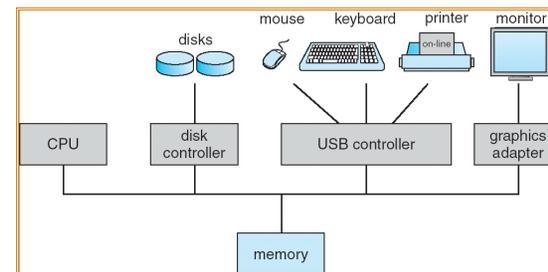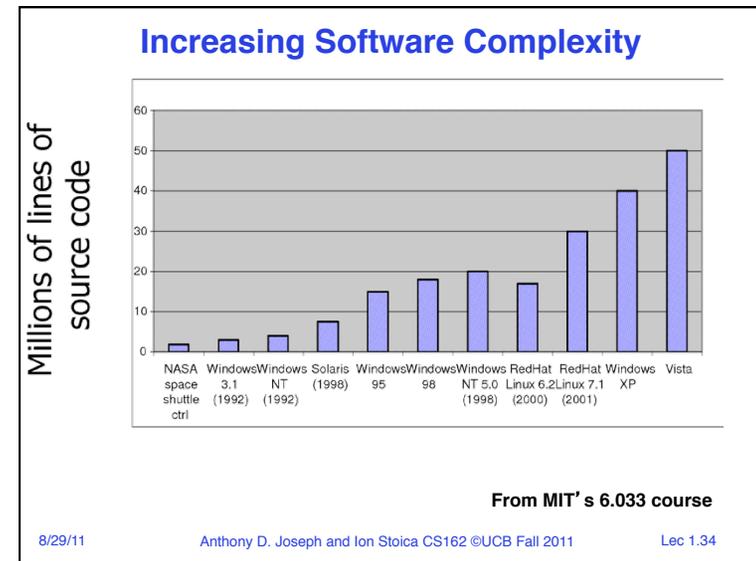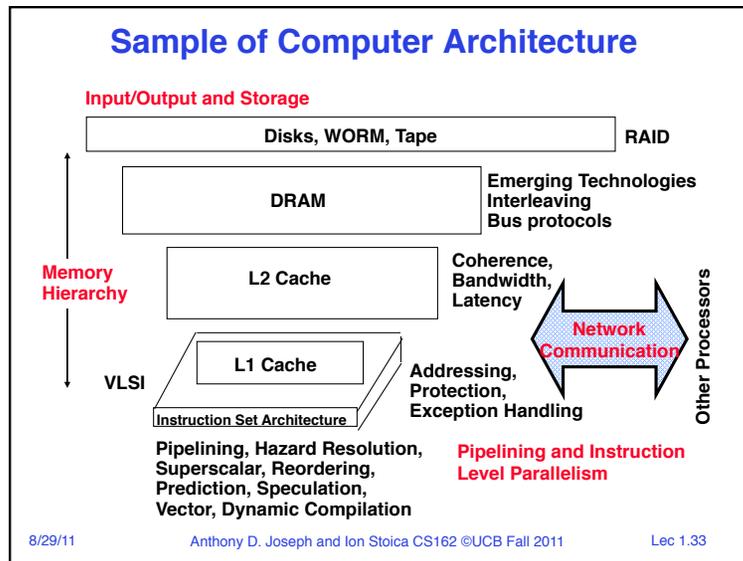---

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles

Page 8

Sample of Computer Architecture

Input/Output and Storage

Disks, WORM, Tape	RAID

DRAM — Emerging Technologies, Interleaving, Bus protocols

Memory Hierarchy

L2 Cache — Coherence, Bandwidth, Latency

Network Communication

Other Processors

VLSI

L1 Cache — Addressing, Protection, Exception Handling

Instruction Set Architecture

Pipelining, Hazard Resolution, Superscalar, Reordering, Prediction, Speculation, Vector, Dynamic Compilation

Pipelining and Instruction Level Parallelism

Increasing Software Complexity

Millions of lines of source code (y-axis: 0–60)

Categories: NASA space shuttle ctrl, Windows 3.1 (1992), Windows NT (1992), Solaris (1998), Windows 95, Windows 98, Windows NT 5.0 (1998), RedHat Linux 6.2 (2000), RedHat Linux 7.1 (2001), Windows XP, Vista

From MIT's 6.033 course

Example: Some Mars Rover ("Pathfinder") Requirements

- Pathfinder hardware limitations/complexity:
  - 20Mhz processor, 128MB of DRAM, VxWorks OS
  - cameras, scientific instruments, batteries, solar panels, and locomotion equipment
  - Many independent processes work together
- Can't hit reset button very easily!
  - Must reboot itself if necessary
  - Must always be able to receive commands from Earth
- Individual Programs must not interfere
  - Suppose the MUT (Martian Universal Translator Module) buggy
  - Better not crash antenna positioning software!
- Further, all software may crash occasionally
  - Automatic restart with diagnostics sent to Earth
  - Periodic checkpoint of results saved?
- Certain functions time critical:
  - Need to stop before hitting something
 - Must track orbit of Earth for communication

How do we tame complexity?

- Every piece of computer hardware different
  - Different CPU
    - » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers, touch screen
  - Different networking environment
    - » Cable, DSL, Wireless, Firewalls,…
- Questions:
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?

Virtual Machine Abstraction

Application

Virtual Machine Interface

Operating System

Hardware

Physical Machine Interface
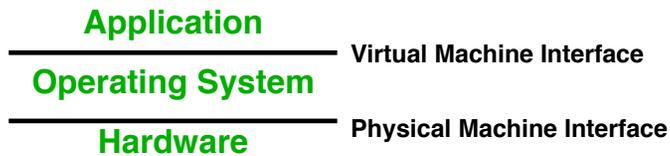
- Software Engineering Problem:
  - Turn hardware/software quirks ⇒ what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc…
- For any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What's the hardware interface? (physical reality)
  - What's the application interface? (nicer abstraction)

Virtual Machines

- Software emulation of an abstract machine
  - Make it look like hardware has features you want
  - Programs from one hardware & OS on another one
- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different Devices appear to have same interface
  - Device Interfaces more powerful than raw hardware
    » Bitmapped display ⇒ windowing system
    » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)
- Fault Isolation
  - Processes unable to directly impact other processes
  - Bugs cannot crash whole machine
- Protection and Portability
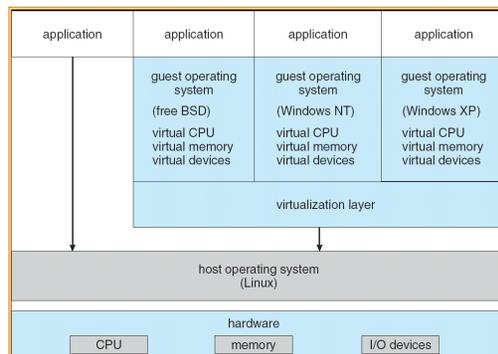 - Java interface safe and stable across many platforms

Virtual Machines (con't): Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs

application	application	application	application
	guest operating system (free BSD) virtual CPU virtual memory virtual devices	guest operating system (Windows NT) virtual CPU virtual memory virtual devices	guest operating system (Windows XP) virtual CPU virtual memory virtual devices
	virtualization layer		

host operating system (Linux)

hardware — CPU — memory — I/O devices

What does an Operating System do?

- Silberschatz and Gavin: "An OS is Similar to a government"
  - Begs the question: does a government do anything useful by itself?

- Coordinator and Traffic Cop:
  - Manages all resources
  - Settles conflicting requests for resources
  - Prevent errors and improper use of the computer

- Facilitator ("useful" abstractions):
  - Provides facilities/services that everyone needs
  - Standard Libraries, Windowing systems
  - Make application programming easier, faster, less error-prone

- Some features reflect both tasks:
  - File system is needed by everyone (Facilitator) …
  - … but File system must be protected (Traffic Cop)

Page 10

What is an Operating System,… Really?

- Most Likely:
  - Memory Management
  - I/O Management
  - CPU Scheduling
  - Synchronization / Mutual exclusion primitives
  - Communications? (Does Email belong in OS?)
  - Multitasking/multiprogramming?

- What about?
  - File System?
  - Multimedia Support?
  - User Interface?
  - Internet Browser? ☺

- Is this only interesting to Academics??

Operating System Definition (Cont'd)

- No universally accepted definition

- "*Everything a vendor ships when you order an operating system*" is good approximation
  - But varies wildly

- "*The one program running at all times on the computer*" is the kernel
  - Everything else is either a system program (ships with the operating system) or an application program

Summary

- Operating systems provide a virtual machine abstraction to handle diverse hardware

- Operating systems coordinate resources and protect users from each other

- Operating systems simplify application development by providing standard services and abstractions

- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery

- CS162 combines things from many other areas of computer science:
  - Languages, data structures, hardware, and algorithms