CS162
Operating Systems and Systems Programming
Lecture 1

What is an Operating System?

January 19th, 2010
Ion Stoica
http://inst.eecs.berkeley.edu/~cs162

Who am I?

Ion Stoica

- Research
  - Networking
    - » Topics: Quality of service, architectures
    - » Projects: Internet Indirection Infrastructure, Declarative Networks
  - Peer-to-Peer
    - » Topics: distributed hash tables, lookup services
    - » Projects: Chord, Internet Indirection Infrastructure
  - Cloud computing
    - » Topics: Scheduling, resource management
    - » Projects: Nexus (Cloud OS), Spark
  - Debugging and Replaying
    - » Projects: Liblog, Friday, ODR (Output Deterministic Replay)

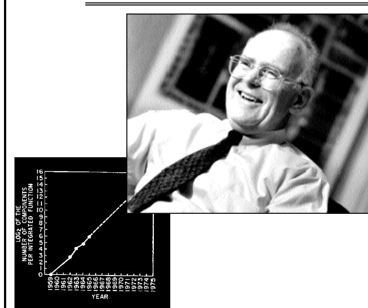
Goals for Today

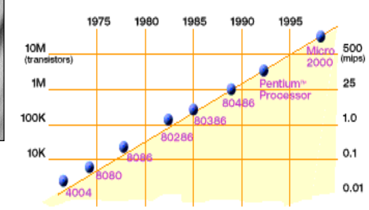
- What is an Operating System?
  - And – what is it not?
- Examples of Operating Systems design
- Why study Operating Systems?
- Oh, and "How does this class operate?"

Interactive is important!
Ask Questions!

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

Technology Trends: Moore's Law





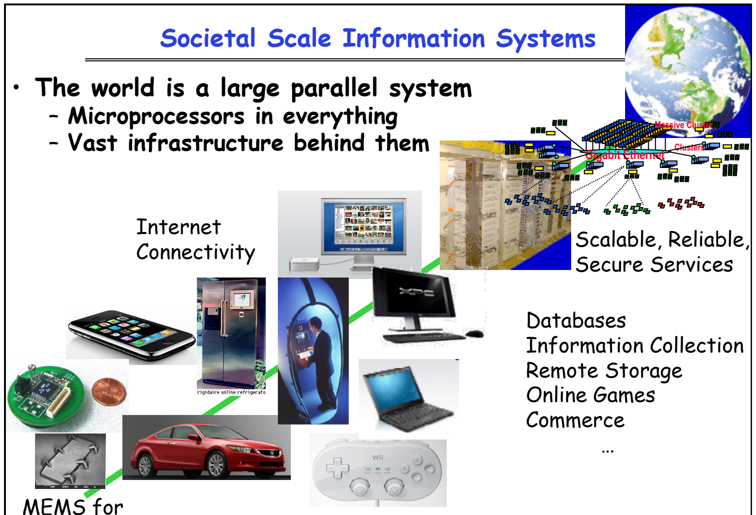
2X transistors/Chip Every 1.5 years
Called "Moore's Law"

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

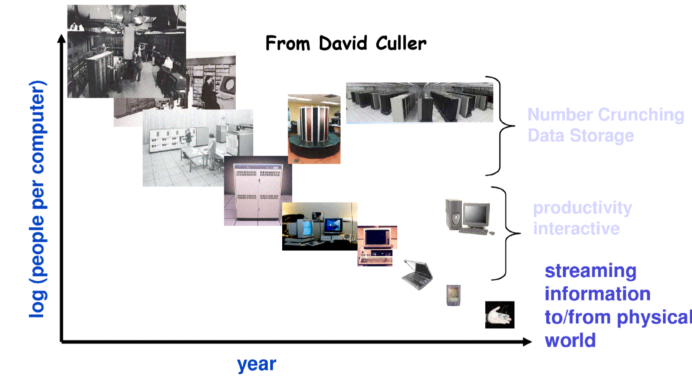
Microprocessors have become smaller, denser, and more powerful.

Societal Scale Information Systems

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

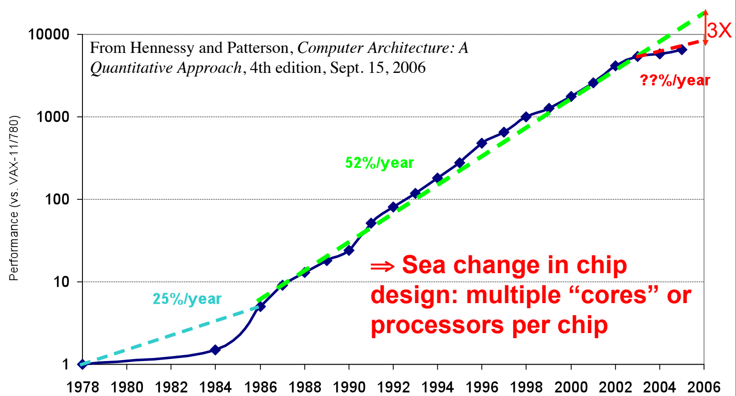


People-to-Computer Ratio Over Time



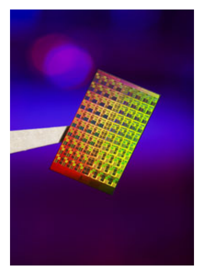
- Today: Multiple CPUs/person!
  - Approaching 100s?

New Challenge: Slowdown in Joy's law of Performance



- 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

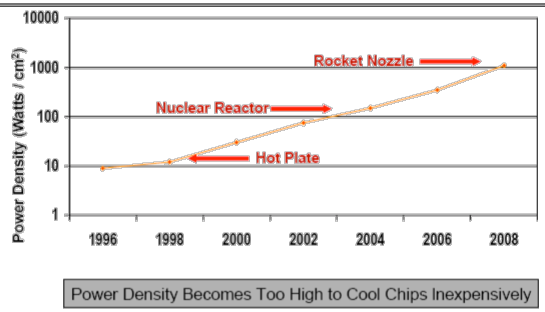


- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two floating point engines /core
  - Mesh-like "network-on-a-chip"
  - 100 million transistors
  - 65nm feature size

Frequency	Voltage	Power	Bandwidth	Performance
3.16 GHz	0.95 V	62W	1.62 Terabits/s	1.01 Teraflops
5.1 GHz	1.2 V	175W	2.61 Terabits/s	1.63 Teraflops
5.7 GHz	1.35 V	265W	2.92 Terabits/s	1.81 Teraflops

- "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

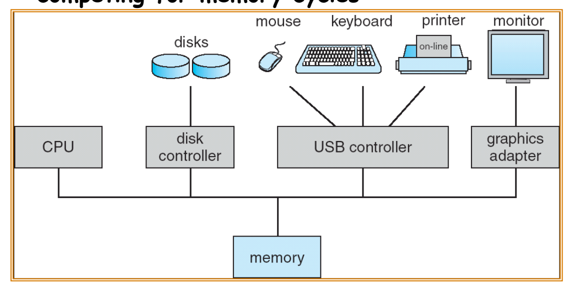
Another Challenge: Power Density



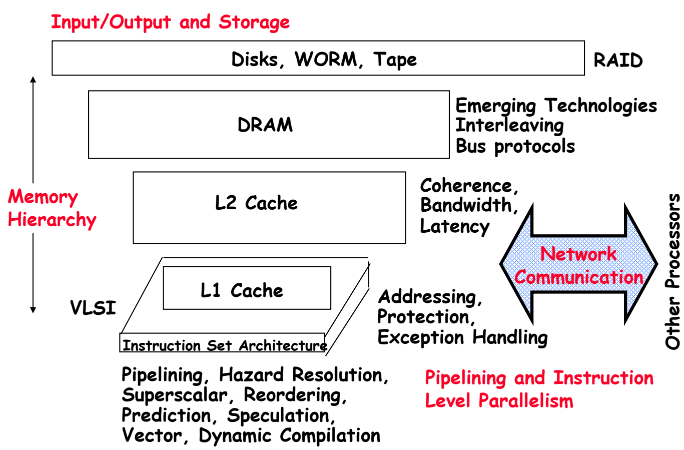
- Moore's Law Extrapolation
  - Potential power density reaching amazing levels!
- Flip side: Battery life very important
  - Moore's law can yield more functionality at equivalent (or less) total energy consumption

Computer System Organization

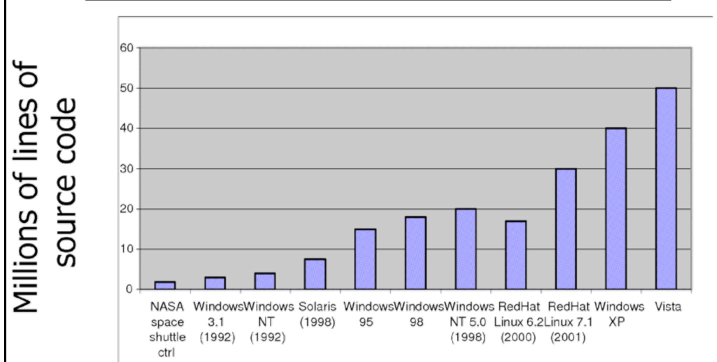
- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Sample of Computer Architecture Topics

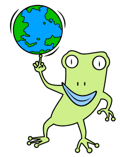


Increasing Software Complexity



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?

OS Tool: Virtual Machine Abstraction

Application

——— Virtual Machine Interface

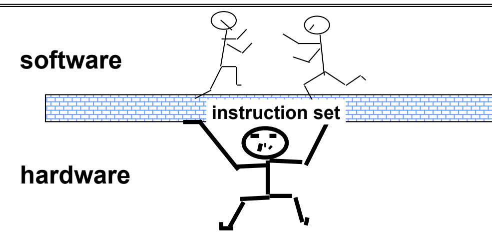
Operating System

——— Physical Machine Interface

Hardware

- 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)

Interfaces Provide Important Boundaries



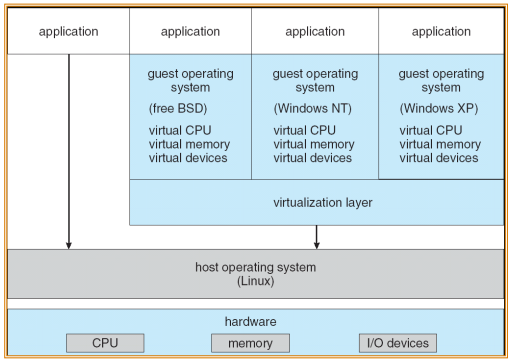
- Why do interfaces look the way that they do?
  - History, Functionality, Stupidity, Bugs, Management
  - CS152 ⇒ Machine interface
  - CS160 ⇒ Human interface
  - CS169 ⇒ Software engineering/management
- Should responsibilities be pushed across boundaries?
  - RISC architectures, Graphical Pipeline Architectures

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



Course Administration

- Instructor: Ion Stoica (istoica@cs.berkeley.edu)
  465 Soda Hall
  Office Hours(Tentative):TT 2:00pm-3:00pm
- TAs: Matei Zaharia (cs162-ta@cory)
  Andy Konwinski (cs162-tb@cory)
  Benjamin Hindman (cs162-tc@cory)
- Labs: Second floor of Soda Hall
- Website: http://inst.eecs.berkeley.edu/~cs162
  Webcast: http://webcast.berkeley.edu/courses/index.php
- Newsgroup: ucb.class.cs162 (use news.csua.berkeley.edu)
- Course Email: cs162@cory.cs.berkeley.edu
- Reader: TBA (Stay tuned!)

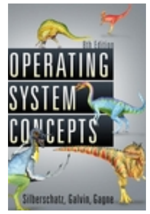
Class Schedule

- Class Time: TT 3:30-5:00 PM, 277 Cory Hall
  - Please come to class. Lecture notes do not have everything in them. The best part of class is the interaction!
  - Also: 5% of the 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
  - No sections this week (obviously); start next week

Section	Time	Location	TA
101	W 10:00A-11:00A	2 Evans	Matei Zaharia
102	W 2:00P-3:00P	75 Evans	Andy Kowinski
103	W 3:00P-4:00P	2 Evans	Ben Hindman

Textbook

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



- Online supplements
  - See "Information" link on course website
  - Includes Appendices, sample problems, etc
- Question: need 8th edition?
  - No, but has new material that we may cover
  - Completely reorganized
  - Will try to give readings from both the 7th and 8th editions on the lecture page

Topic Coverage

Textbook: Silberschatz, Galvin, and Gagne, *Operating Systems Concepts*, 8th Ed., 2008

- 1 week: Fundamentals (Operating Systems Structures)
- 1.5 weeks: Process Control and Threads
- 2.5 weeks: Synchronization and scheduling
- 2 week: Protection, Address translation, Caching
- 1 week: Demand Paging
- 1 week: File Systems
- 2.5 weeks: Networking and Distributed Systems
- 1 week: Protection and Security
- ??: Advanced topics

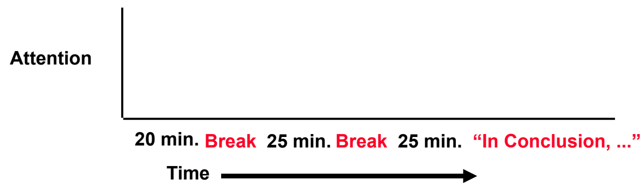
Grading

- Rough Grade Breakdown
  - One Midterm: 20% each
    One Final: 25%
    Four Projects: 50% (i.e. 12.5% each)
    Participation: 5%
- Four Projects:
  - Phase I: Build a thread system
  - Phase II: Implement Multithreading
  - Phase III: Caching and Virtual Memory
  - Phase IV: Networking and Distributed Systems
- Late Policy:
  - No slip days!
  - 10% off per day after deadline

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!!!
  - Everyone must keep an on-line notebook
- 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?

Typical Lecture Format



- 1-Minute Review
- 20-Minute Lecture
- 5- Minute Administrative Matters
- 25-Minute Lecture
- 5-Minute Break (water, stretch)
- 25-Minute Lecture

Lecture Goal

Interactive!!!

Computing Facilities

- Every student who is enrolled should get an account form at 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 into your new account this week and fill out the questions
- Project Information:
  - See the "Projects and Nachos" link off the course home page
- Newsgroup (ucb.class.cs162):
  - Read this regularly!

What does an Operating System do?

- Silerschatz 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:
  - Provides facilities that everyone needs
  - Standard Libraries, Windowing systems
  - Make application programming easier, faster, less error-prone
- Some features reflect both tasks:
  - E.g. File system is needed by everyone (Facilitator)
  - But File system must be Protected (Traffic Cop)

What is an Operating System,… Really?

- Most Likely:
  - Memory Management
  - I/O Management
  - CPU Scheduling
  - 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.)

- 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

OS Systems Principles

- OS as illusionist:
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- OS as government:
  - Protect users from each other
  - Allocate resources efficiently and fairly
- OS as complex system:
  - Constant tension between simplicity and functionality or performance
- OS as history teacher
  - Learn from past
  - Adapt as hardware tradeoffs change

Why Study Operating Systems?

- Learn how to build complex systems:
  - How can you manage complexity for future projects?
- Engineering issues:
  - Why is the web so slow sometimes? Can you fix it?
  - What features should be in the next mars Rover?
  - How do large distributed systems work? (Bittorrent, etc)
- Buying and using a personal computer:
  - Why different PCs with same CPU behave differently
  - How to choose a processor (Opteron, Itanium, Celeron, Pentium)
  - Should you get Windows XP, 2000, Linux, Mac OS …?
- Business issues:
  - Should your division buy thin-clients vs PC?
- Security, viruses, and worms
  - What exposure do you have to worry about?

"In conclusion…"

- 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
- 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