COS 318: Operating Systems

Introduction

Andy Bavier
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall10/cos318/

Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS318?

Course Staff and Logistics

- Instructor
  - Andy Bavier, 212 CS Building, acb@cs.princeton.edu
    Office hours: Tue 3-5pm
- Teaching Assistants
  - Prem Gopalan, pgopalan@cs.princeton.edu
    Office hours: Fri 10am-noon
  - Dominic Kao, dkthree@cs.princeton.edu
    Office hours: Fri 11am-1pm
- Information
  - Website:
    - http://www.cs.princeton.edu/courses/archive/fall10/cos318/
  - **Subscribe to cos318@lists.cs.princeton.edu**

Resolve “TBD”

- Regular precept
  - Time: Tuesday 7:30pm – 8:30pm
  - Location: default is this room, CS 105

- Review of x86 Real-Mode Assembly
  - Monday Sep. 20, 7:30pm – 8:30pm

- Design review
  - Monday Sep. 27, 5pm -- 9pm
  - Sign up online

COS318 in Systems Course Sequence

- Prerequisites
  - COS 217: Introduction to Programming Systems
  - COS 226: Algorithms and Data Structures
- 300-400 courses in systems
  - **COS318: Operating Systems**
  - COS320: Compiler Techniques
  - COS333: Advanced Programming Techniques
  - COS425: Database Systems
  - COS471: Computer Architecture
- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks

Course Materials

- Textbook
  - *Modern Operating Systems*, 3rd Edition, Andrew S. Tanenbaum
- Lecture notes
  - Available on website
- Precept notes
  - Available on website
- Other resources – on website

Exams, Participation and Grading

- Grading
  - First 5 projects: 50% with extra points
  - Midterm: 20%
  - Final project: 20%
  - Reading & participation: 10%
- Midterm Exam
  - Test lecture materials and projects
  - Tentatively scheduled on Thursday of the midterm week
- Reading and participation
  - Submit your reading notes BEFORE each lecture
  - Sign-in sheet at each lecture
  - Grading (3: excellent, 2: good, 1: poor, 0: none)

The First 5 Projects

- Projects
  - Bootup (150-300 lines)
  - Non-preemptive kernel (200-250 lines)
  - Preemptive kernel (100-150 lines)
  - Interprocess communication and driver (300-350 lines)
  - Android OS (??? lines)
- How
  - Pair up with a partner, will change after 3 projects
  - Each project takes two weeks
  - Design review at the end of week one
  - All projects due Mondays 11:59pm
- The Lab
  - Linux cluster in 010 Friends Center, a good place to be
  - You can setup your own Linux PC to do projects

Project Grading

- Design Review
  - Signup online for appointments
  - 10 minutes with the TA in charge
  - 0-5 points for each design review
  - 10% deduction if missing the appointment
- Project completion
  - 10 points for each project
  - Extra points available
- Late policy of grading projects
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%

Final Project

- A simple file system
- Grading (20 points)
- Do it alone
- Due on Dean's date (~3 weeks)

Things To Do

- **Do not put your code or designs or thoughts on the Web**
  - Other schools are using similar projects
  - Not even on Facebook or the like
- Follow Honor System: ask when unsure, cooperation OK but work is your own (or in pairs for projects)
- For today's material:
  - Read MOS 1.1-1.3
- For next time
  - Read MOS 1.4-1.5

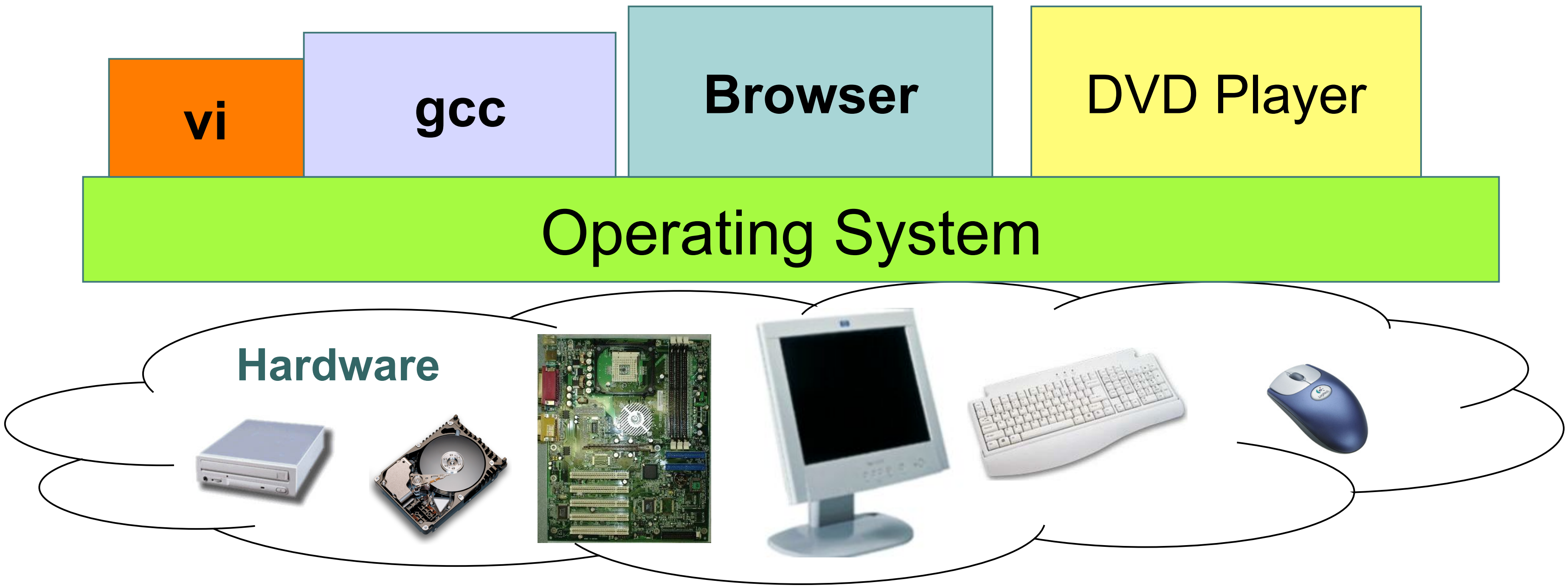
Email to acb@cs.princeton.edu

- Name
- Year
- Major
- Why you're taking the class
- What you'd like to learn

Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS318?

What Is an Operating System?



- Software that sits between applications and hardware
  - Also between different applications and different users
- Has privileged access to hardware
- Provides services and interfaces to applications

User applications call OS routines for access and services

What Does an Operating System Do?

- Provides a *layer of abstraction* for hardware resources
  - Allows user programs to deal with higher-level, simpler, and more portable concepts than the raw hardware
    - E.g., files rather than disk blocks
  - Makes finite resources seem “infinite”
- Manages the resources
  - Manage complex resources and their interactions for an application
  - Allow multiple applications to share resources without hurting one another
  - Allow multiple users to share resources without hurting one another

Abstraction

- How to handle complexity
- Hide underlying details, and provide cleaner, easier-to-use, more elegant concepts and interfaces
  - Also provides standardized interfaces despite diversity of implementation underneath
- A key to understanding Operating Systems
- A key principle in Computer Science

Example of Abstraction: Disk

- Disk hardware and operations are very complex
  - Multiple heads, cylinders, sectors, segments
  - Have to wait for physical movement before writing or reading data to/from disk
  - Data stored discontiguously for performance, reliability
  - To read or write simple data would take a lot of coordination if dealing with the hardware directly
  - Sizes and speeds are different on different computers
- OS provides simple read() and write() calls as the application programmer's interface (API)
  - Manages the complexity transparently, in conjunction with the disk controller hardware

Example of Abstraction: Networks

- Data communicated from one computer to another are:
  - Broken into fragments that are sent separately, and arrive at different times and out of order
  - Waited for and assembled at the destination
  - Sometimes lost, so fragments have to be resent
  - An application programmer doesn't want to manage this
- OS provides a simple send() and receive() interface
  - Takes care of the complexity, in conjunction with the networking hardware

Resource Management

- Allocation
- Virtualization
- Reclamation
- Protection

Resource Allocation

- Computer has finite resources
- Different applications and users compete for them
- OS dynamically manages which applications get how many resources
- *Multiplex* resources in space and time
  - Time multiplexing: CPU, network
  - Space multiplexing: disk, memory

- E.g., what if an application runs an infinite loop?

```
while (1);
```

Resource Virtualization

- OS gives each program the illusion of effectively infinite, private resources
  - "infinite" memory (by backing up to disk)
  - CPU (by time-sharing)

Resource Reclamation

- The OS giveth, and the OS taketh away
  - Voluntary or involuntary at runtime
  - Implied at program termination
  - Cooperative

Protection

- You can’t hurt me, I can’t hurt you
- OS provides safety and security
- Protects programs and their data from one another, as well as users from one another
- E.g., what if I could modify your data, either on disk or while your program was running?

Mechanism vs. policy

- Mechanisms are tools or vehicles to implement policies
- Examples of policies:
  - All users should be treated equally
  - Preferred users should be treated better

Today

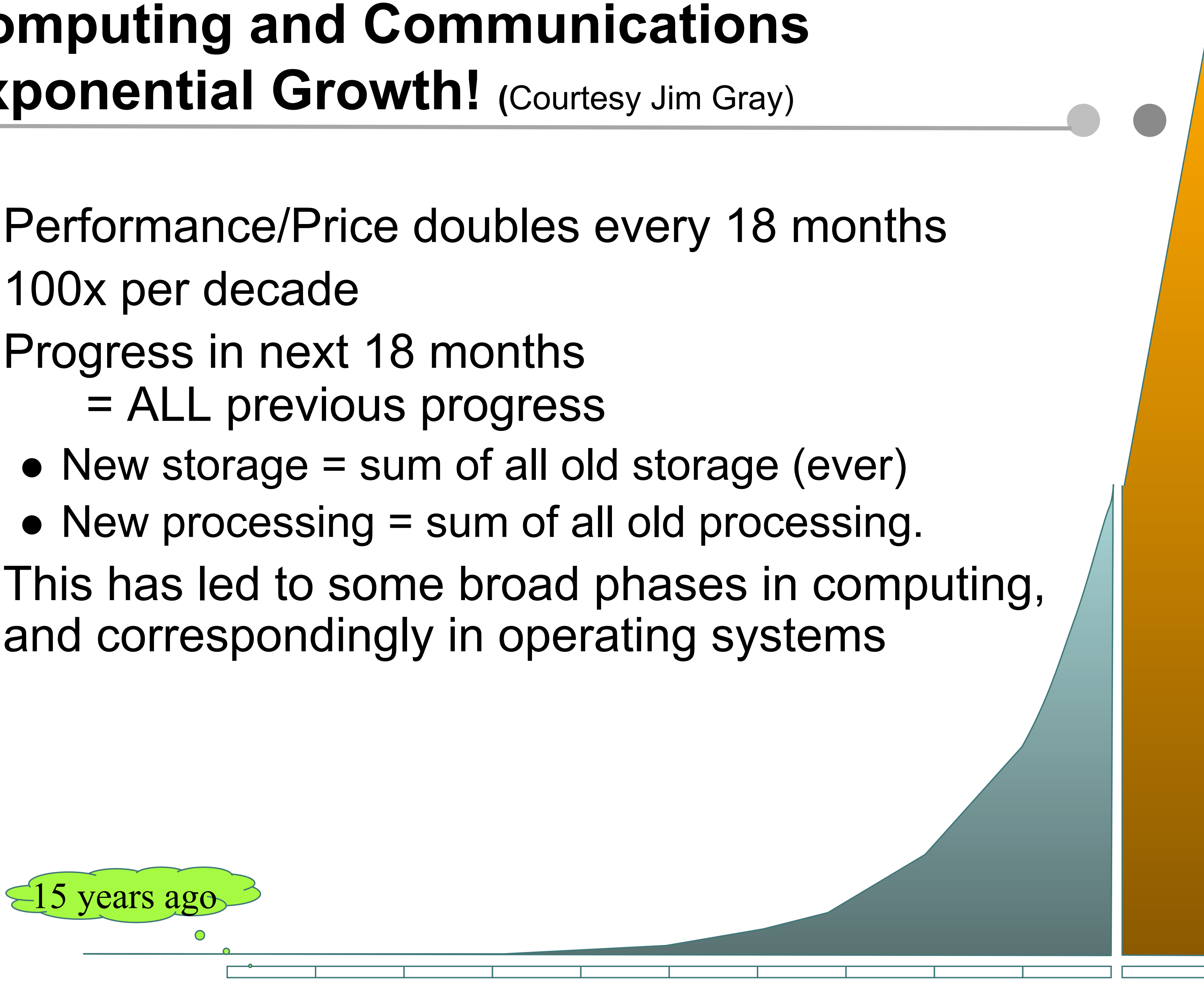
- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS318?

A Typical Academic Computer (1988 vs. 2008)

	1988	2008	Ratio
Intel CPU transistors	0.5M	1.9B	~4000x
Intel CPU core x clock	10Mhz	4×2.66Ghz	~1000x
DRAM	2MB	16GB	8000x
Disk	40MB	1TB	25,000x
Network BW	10Mbits/sec	10GBits/sec	1000x
Address bits	32	64	2x
Users/machine	10s	< 1	>10x
$/machine	$30K	$3K	1/10x
$/Mhz	$30,000/10	$3,000/10,000	1/10,000x

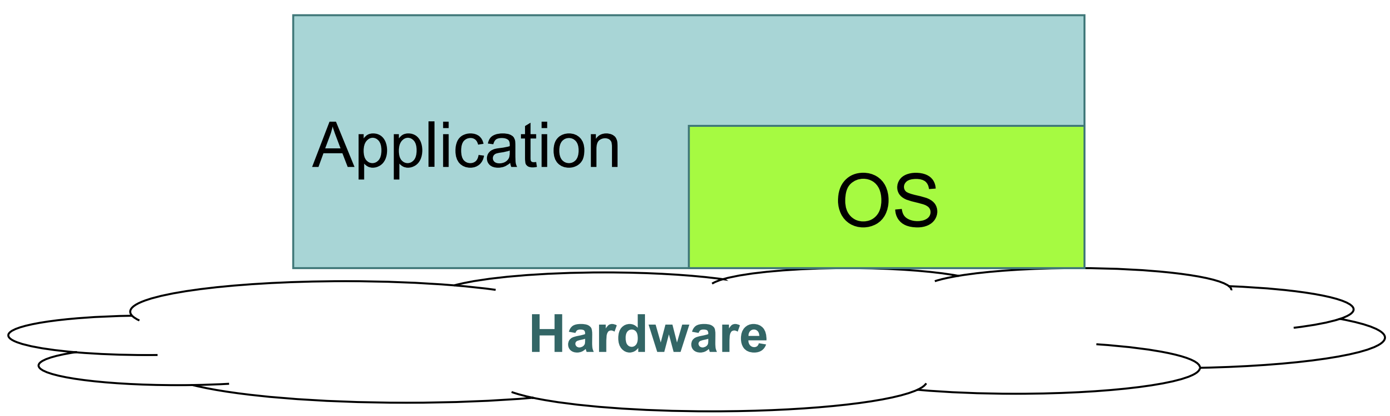
Computing and Communications Exponential Growth! (Courtesy Jim Gray)

- Performance/Price doubles every 18 months
- 100x per decade
- Progress in next 18 months
  = ALL previous progress
  - New storage = sum of all old storage (ever)
  - New processing = sum of all old processing.
- This has led to some broad phases in computing, and correspondingly in operating systems



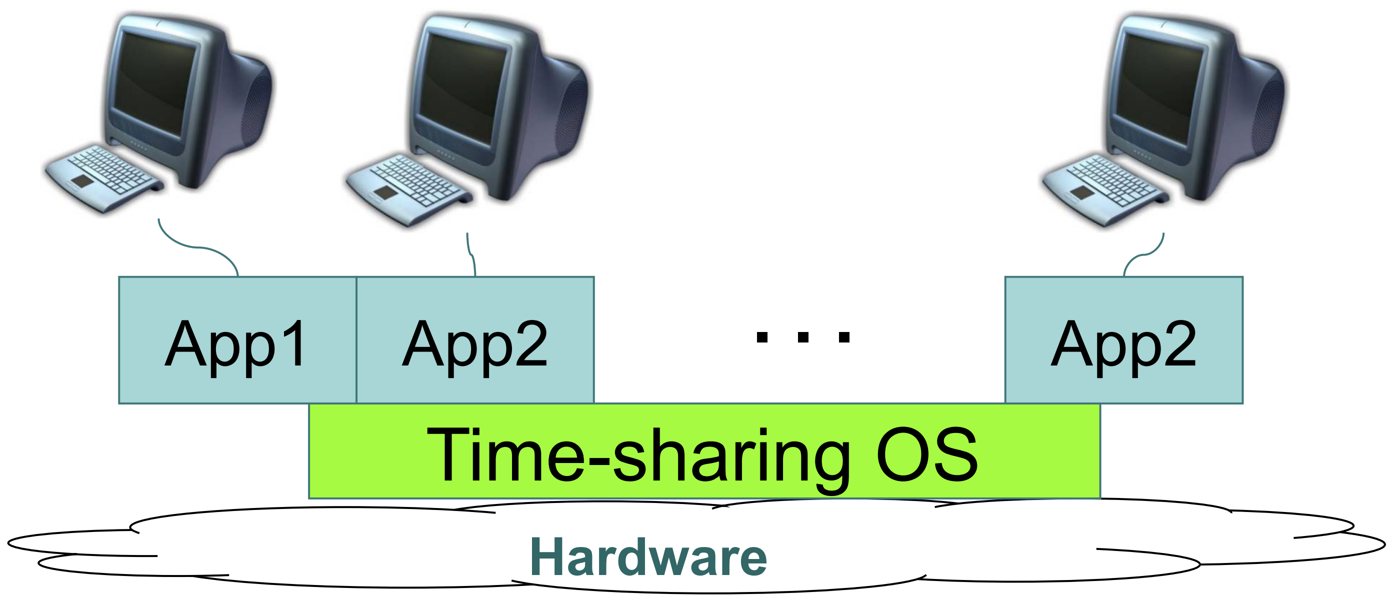
Phase 1: Batch Systems

- Hardware very expensive, only one user at a time
- Batch processing: load, run, print
  - OS linked in as a subroutine library
- Problem: better system utilization
  - System idle when job waiting for I/O
- Development: multiprogramming
  - Multiple jobs resident in computer's memory
  - Hardware switches between them (interrupts)
  - Memory protection: keep bugs to individual programs



Phase 2: Time Sharing

- Problem: batch jobs hard to debug
- Use cheap terminals to share a computer interactively
- MULTICS: designed in 1963, run in 1969
- Shortly after, Unix enters the mainstream
- Issue: thrashing as the number of users increases



Phase 3: Personal Computer

- Personal computer
  - Altos OS, Ethernet, Bitmap display, laser printer
  - Pop-menu window interface, email, publishing SW, spreadsheet, FTP, Telnet
  - Eventually >100M units per year
- PC operating system
  - Memory protection
  - Multiprogramming
  - Networking

Now: > 1 Machines per User

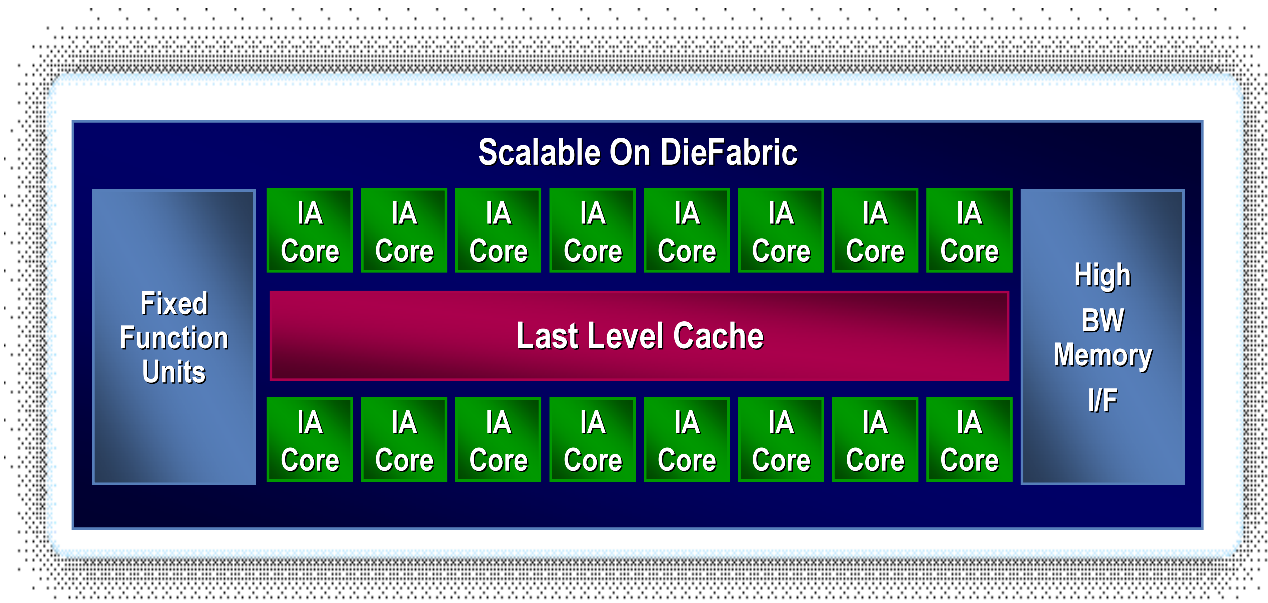
- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle
- OS are specialized
  - Embedded OS
  - Specially configured general-purpose OS

Now: Multiple Processors per Machine

- Multiprocessors
  - SMP: Symmetric MultiProcessor
  - ccNUMA: Cache-Coherent Non-Uniform Memory Access
  - General-purpose, single-image OS with multiproccesor support
- Multicomputers
  - Supercomputer with many CPUs and high-speed communication
  - Specialized OS with special message-passing support
- Clusters
  - A network of PCs
  - Commodity OS

Trend: Multiple "Cores" per Processor

- Multicore or Manycore transition
  - Intel and AMD have released 4-core CPUs
  - SUN's Niagara processor has 8-cores
  - Azul packed 24-cores onto the same chip
  - Intel has a TFlop-chip with 80 cores
- Accelerated need for software support
  - OS support for manycores
  - Parallel programming of applications



Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS318?

Why Study OS?

- OS is a key part of a computer system
  - It makes our life better (or worse)
  - It is “magic” to realize what we want
  - It gives us “power”
- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware
  - Best way to learn concurrent programming
- Understand how a system works
  - How many procedures does a key stroke invoke?
  - What happens when your application references 0 as a pointer?
  - Building a small OS will go a long way…

Why Study OS?

- Important for studying other areas
  - Networking, distributed systems, …
- Full employment
  - New hardware capabilities and organizations
  - New features
  - New approaches
  - Engineering tradeoffs keep shifting as the hardware changes below and the apps change above

Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS318?

What's in COS 318?

- Methodology
  - Lectures with discussions
  - Readings with topics
  - Six projects to build a small but real OS, play with Android
- Covered concepts
  - Operating system structure
    - Processes, threads, system calls and virtual machine monitor
  - Synchronization
    - Mutex, semaphores and monitors
  - I/O subsystems
    - Device drivers, IPC, and introduction to networking
  - Virtual memory
    - Address spaces and paging
  - Storage system
    - Disks and file system

What is COS 318 Like?

- Is it theoretical or practical?
  - Focus on concepts, also getting hands dirty in projects
  - Engineering tradeoffs: requirements, constraints, optimizations, imperfections
  - High rate of change in the field yet lots of inertia in OSs
- Is it easy?
  - No. Fast-paced, hard material, a lot of programming
- What will help me succeed?
  - Solid C background, pre-reqs, tradeoff thinking
  - NOT schedule overload