ICS 143 - Principles of Operating Systems

Lecture 1 - Introduction and Overview

T,Th 3:30 - 4:50 p.m.

Prof. Nalini Venkatasubramanian

(nalini@ics.uci.edu)

[lecture slides contains some content adapted from : Silberschatz textbook authors, John Kubiatowicz (Berkeley)]

ICS 143 Winter 2012 Staff

Instructor:

Prof. Nalini Venkatasubramanian (Venkat)

(nalini@ics.uci.edu)

Teaching Assistants:

Daniel Miller(djmiller@uci.edu)

Readers

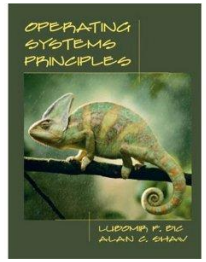
Santanu Sarma(santanus@uci.edu)

Mehdi Sadri (msadri@uci.edu)

Course logistics and details

- Course Web page -
  - http://www.ics.uci.edu/~ics143
- Lectures - TTh 3:30-4:50p.m, DBH 1100
- Discussions – F 12:00-12:50 p.m, EH 1200
- ICS 143 Textbook:

  Operating System Concepts -- Eighth Edition
  Silberschatz and Galvin, Addison-Wesley Inc.
  (Seventh,Sixth and Fifth editions, and Java Versions are fine as well).



- Alternate Book
  - Principles of Operating Systems, L.F. Bic and A.C. Shaw, Prentice-Hall/Pearson Education, 2003. ISBN 0130266116.

Course logistics and details

- Homeworks and Assignments
  - 4 written homeworks in the quarter
  - 1 programming assignment (knowledge of C++ or Java required).
    - Handed out at midterm; submit/demo during Finals Week
    - Multistep assignment – don't start in last week of classes!!!
  - Late homeworks will not be accepted.
  - All submissions will be made using the EEE Dropbox for the course
- Tests
  - Midterm - tentatively Tuesday, Week 6 (Feb 14th) in class
  - Final Exam - as per UCI course catalog, March 20th (4-6 p.m.)

ICS 143 Grading Policy

- Homeworks - 30%
  - 4 written homeworks each worth 5% of the final grade.
  - 1 programming assignment worth 10% of the final grade
- Midterm - 30% of the final grade
- Final exam - 40% of the final grade

- Final assignment of grades will be based on a curve.

Lecture Schedule

- Week 1:
  - Introduction to Operating Systems, Computer System Structures, Operating System Structures
- Week 2 : Process Management
  - Processes and Threads, CPU Scheduling
- Week 3: Process Management
  - CPU Scheduling, Process Synchronization
- Week 4: Process Management
  - Process Synchronization
- Week 5: Process Management
  - Deadlocks

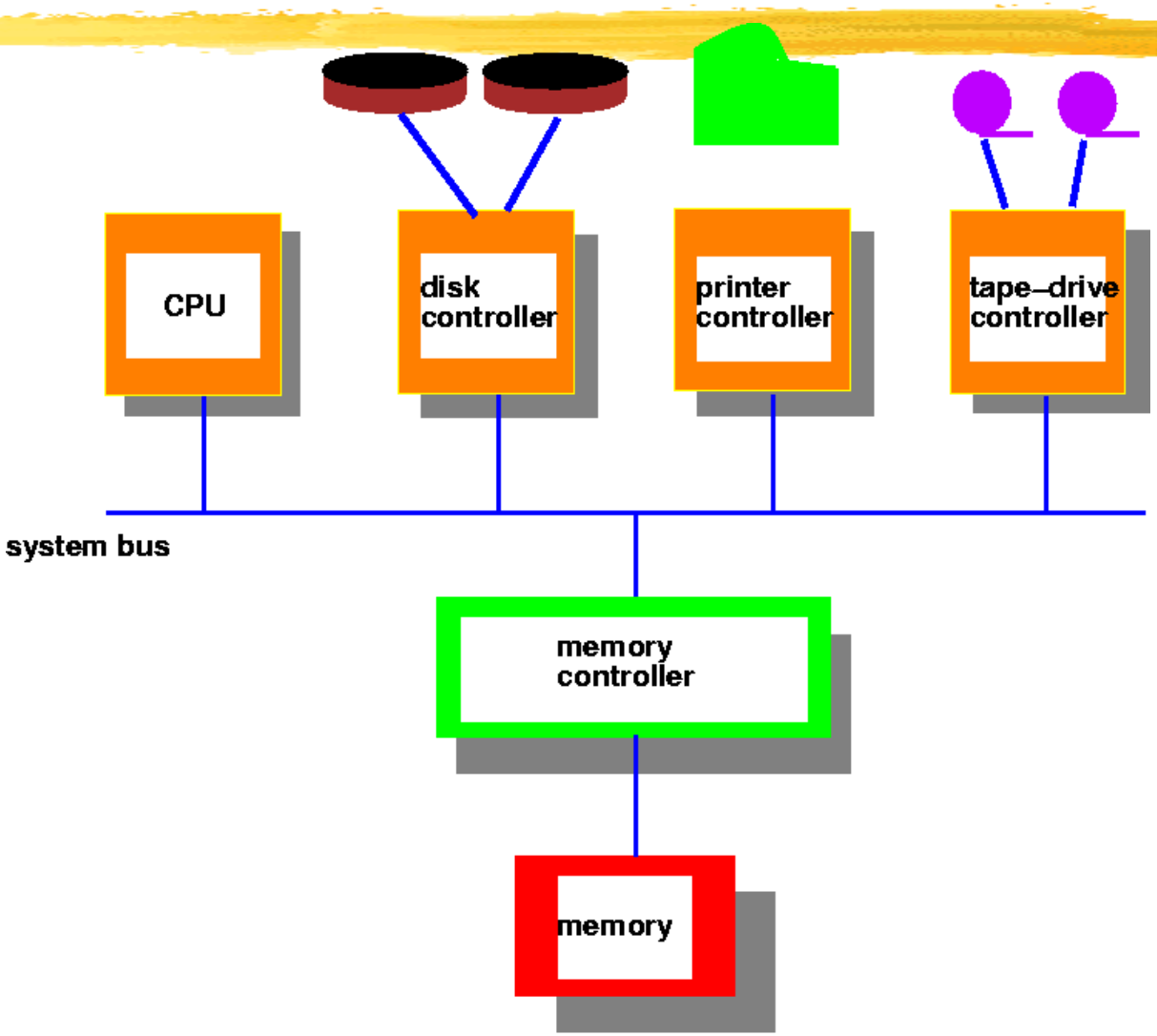
Course Schedule

- Week 6 - Storage Management
  - Midterm exam, Memory Management
- Week 7 - Storage Management
  - Memory Mangement, Virtual Memory
- Week 8 - I/O Systems
  - Virtual Memory, Filesystem Interface,
- Week 9 - Other topics
  - FileSystems Implementation, I/O subsystems
- Week 10 - Other topics
  - Case study – UNIX, WindowsNT, course revision and summary.

Introduction

- What is an operating system?
- Early Operating Systems
  - Simple Batch Systems
  - Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems

Computer System Architecture



What is an Operating System?

- An OS is a program that acts an intermediary between the user of a computer and computer hardware.
- Major cost of general purpose computing is software.
  - OS simplifies and manages the complexity of running application programs efficiently.

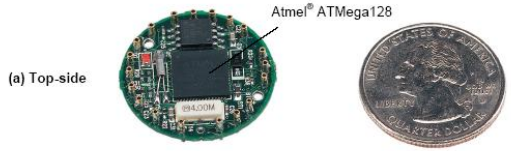
Goals of an Operating System

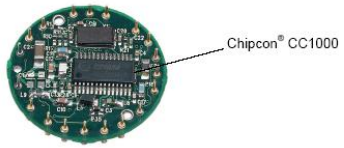
- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
  - Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability
  - error confinement, fault tolerance, reconfiguration.

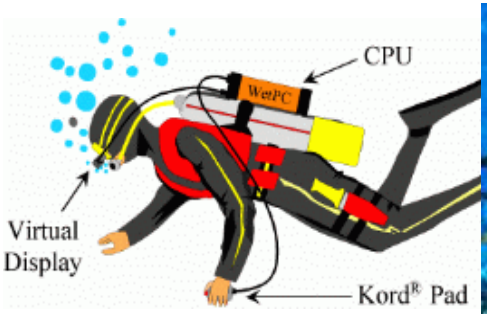
Why should I study Operating Systems?

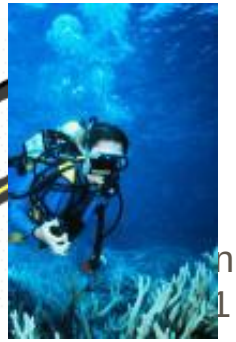
- Need to understand interaction between the hardware and applications
  - New applications, new hardware..
  - Inherent aspect of society today
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Increasing need for specialized operating systems
  - e.g. embedded operating systems for devices - cell phones, sensors and controllers
  - real-time operating systems - aircraft control, multimedia services

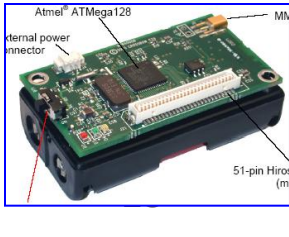
Systems Today









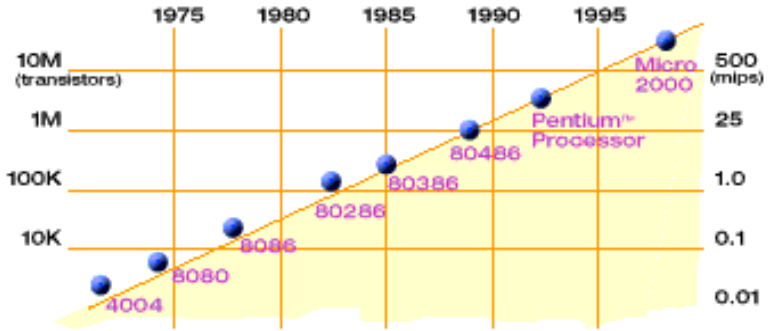


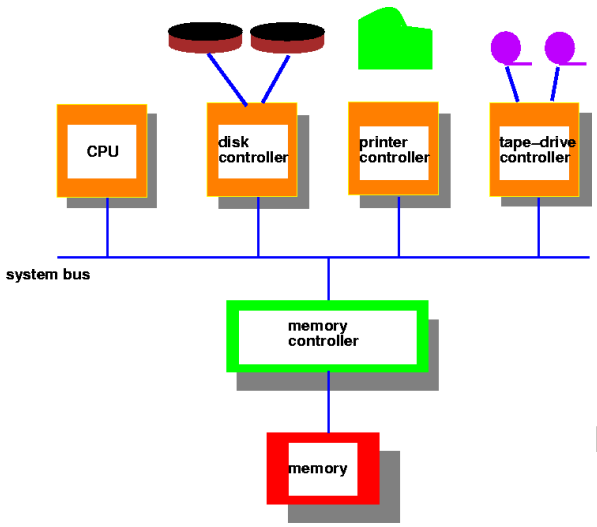
Irvine Sensorium

Hardware Complexity Increases

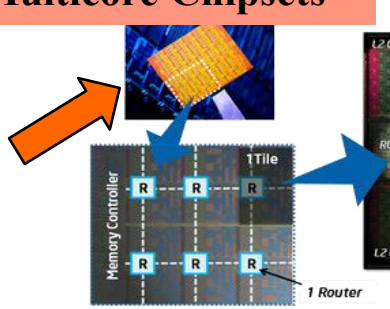
Moore's Law: 2X transistors/Chip Every 1.5 years

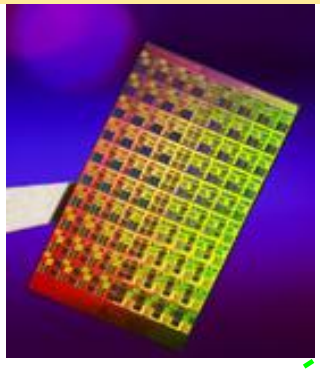
From Berkeley OS course

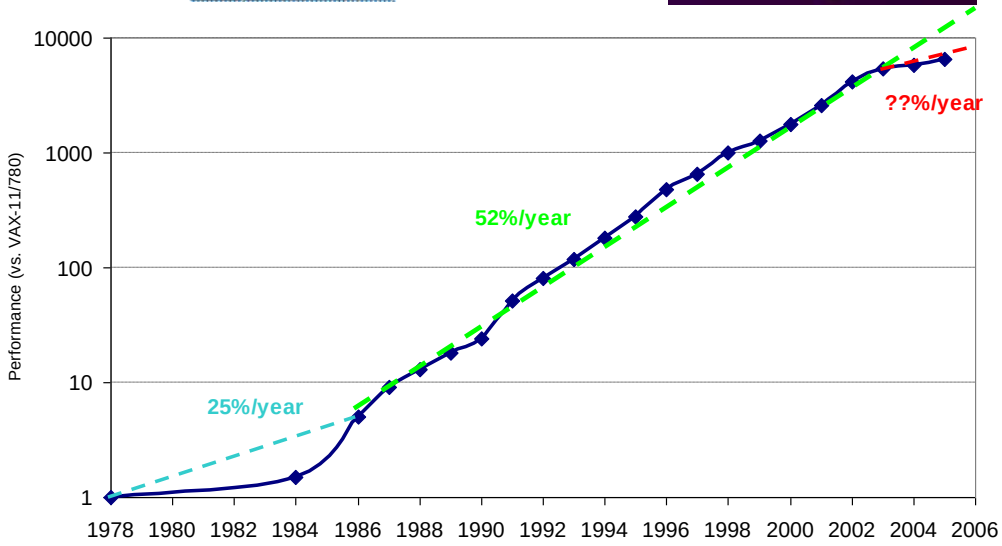




Intel Multicore Chipsets

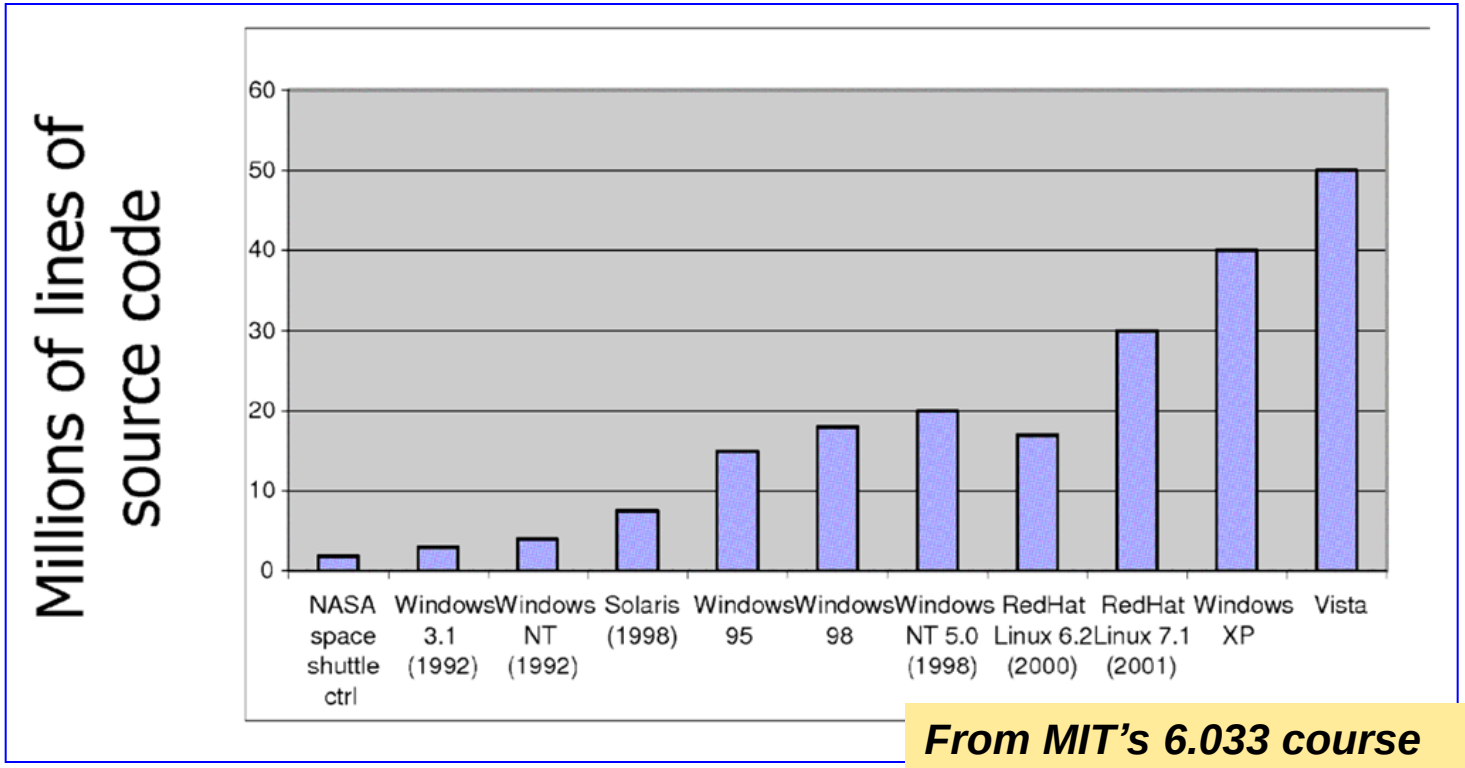






From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

Software Complexity Increases

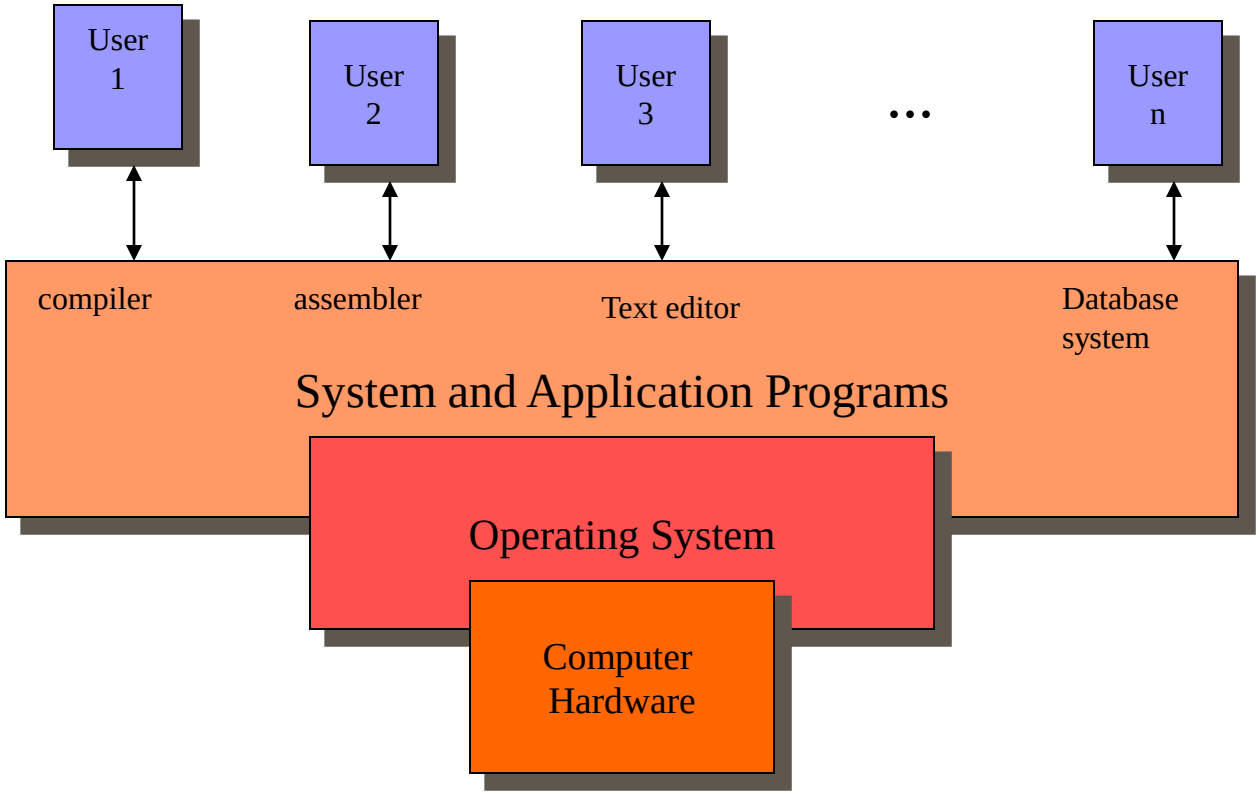


From MIT's 6.033 course

Computer System Components

- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices).
- Operating System
  - Controls and coordinates the use of hardware among application programs.
- Application Programs
  - Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).
- Users
  - People, machines, other computers

Abstract View of System



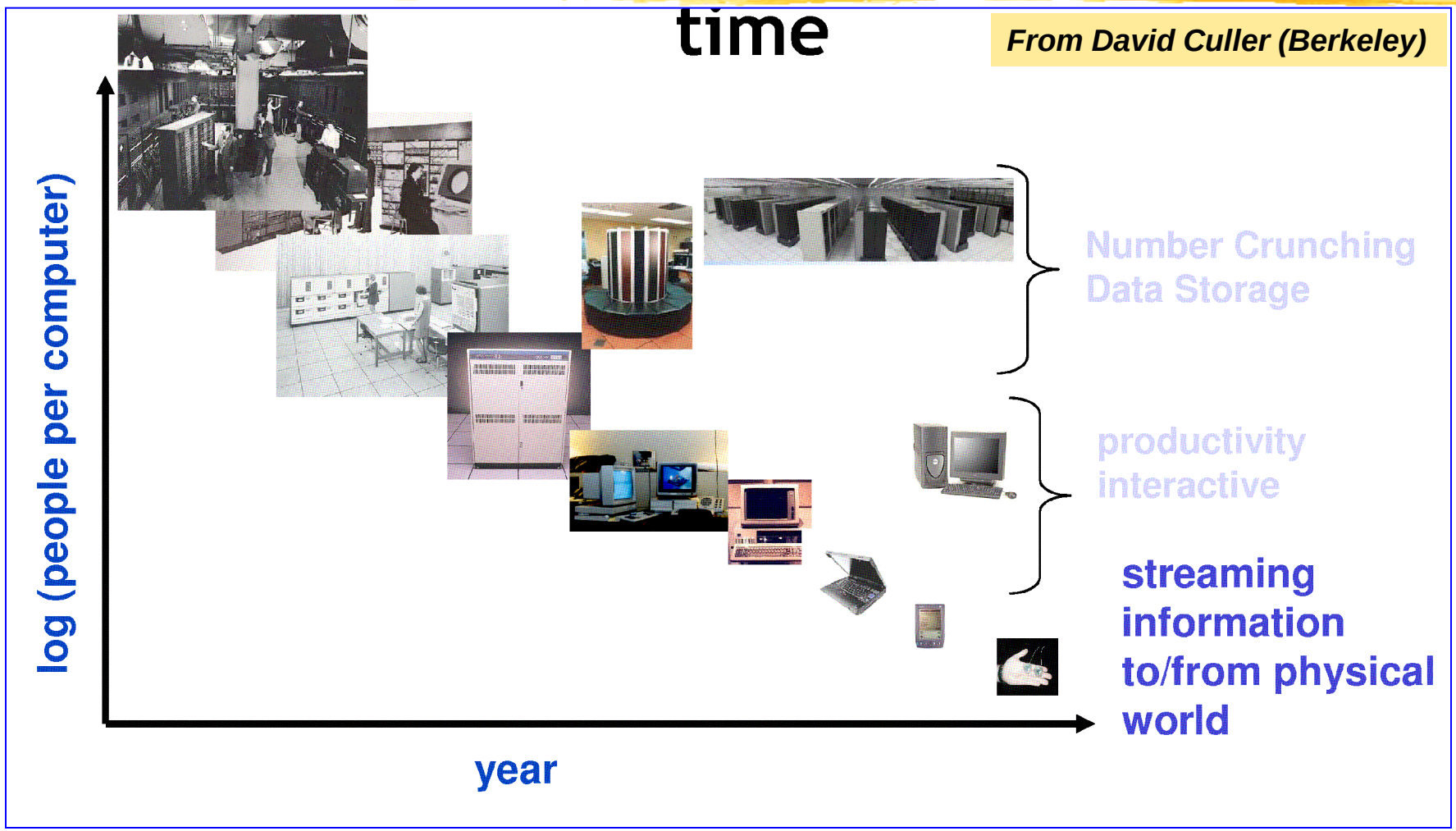
Operating System Views

- Resource allocator
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program
  - Controls execution of user programs and operation of I/O devices.
- Kernel
  - The program that executes forever (everything else is an application with respect to the kernel).

Operating System Spectrum

- Monitors and Small Kernels
  - special purpose and embedded systems, real-time systems
- Batch and multiprogramming
- Timesharing
  - workstations, servers, minicomputers, timeframes
- Transaction systems
- Personal Computing Systems
- Mobile Platforms, devices (of all sizes)

People-to-Computer Ratio Over Time



Early Systems - Bare Machine (1950s)

Hardware – *expensive* ; Human – *cheap*

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
- Secure execution
- Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

From John Ousterhout slides

Simple Batch Systems (1960's)

- Reduce setup time by batching jobs with similar requirements.
- Add a card reader, Hire an operator
  - User is NOT the operator
  - Automatic job sequencing
    - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job and then back to monitor.
  - Problem
    - Need to distinguish job from job and data from program.

From John Ousterhout slides

Supervisor/Operator Control

- Secure monitor that controls job processing
  - Special cards indicate what to do.
  - User program prevented from performing I/O
- Separate user from computer
  - User submits card deck
  - cards put on tape
  - tape processed by operator
  - output written to tape
  - tape printed on printer
- Problems
  - Long turnaround time - up to 2 DAYS!!!
  - Low CPU utilization
    - I/O and CPU could not overlap; slow mechanical devices.

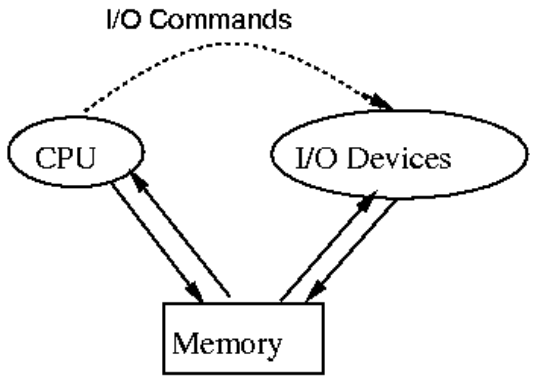
From John Ousterhout slides

Batch Systems - Issues

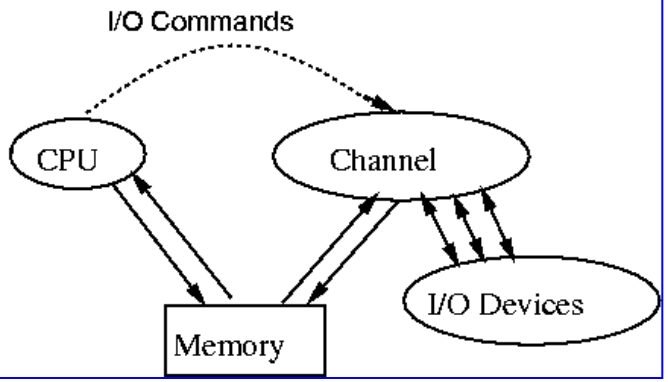
- Solutions to speed up I/O:
- Offline Processing
  - load jobs into memory from tapes, card reading and line printing are done offline.
- Spooling
  - Use disk (random access device) as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces notion of a job pool that allows OS choose next job to run so as to increase CPU utilization.

Speeding up I/O

- Direct Memory Access (DMA)



- Channels



Batch Systems - I/O completion

- How do we know that I/O is complete?
  - Polling:
    - Device sets a flag when it is busy.
    - Program tests the flag in a loop waiting for completion of I/O.
  - Interrupts:
    - On completion of I/O, device forces CPU to jump to a specific instruction address that contains the interrupt service routine.
    - After the interrupt has been processed, CPU returns to code it was executing prior to servicing the interrupt.

Multiprogramming

- Use interrupts to run multiple programs simultaneously
  - When a program performs I/O, instead of polling, execute another program till interrupt is received.
- Requires secure memory, I/O for each program.
- Requires intervention if program loops indefinitely.
- Requires CPU scheduling to choose the next job to run.

Timesharing

Hardware – ***getting cheaper***; **Human** – ***getting expensive***

- Programs queued for execution in FIFO order.
- Like multiprogramming, but timer device interrupts after a quantum (timeslice).
  - Interrupted program is returned to end of FIFO
  - Next program is taken from head of FIFO
- Control card interpreter replaced by command language interpreter.

Timesharing (cont.)

- Interactive (action/response)
  - when OS finishes execution of one command, it seeks the next control statement from user.
- File systems
  - online filesystem is required for users to access data and code.
- Virtual memory
  - Job is swapped in and out of memory to disk.

Personal Computing Systems

Hardware – *cheap* ; Human – *expensive*

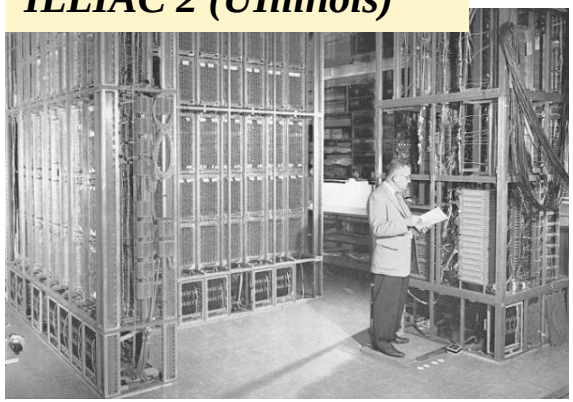
- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

Parallel Systems

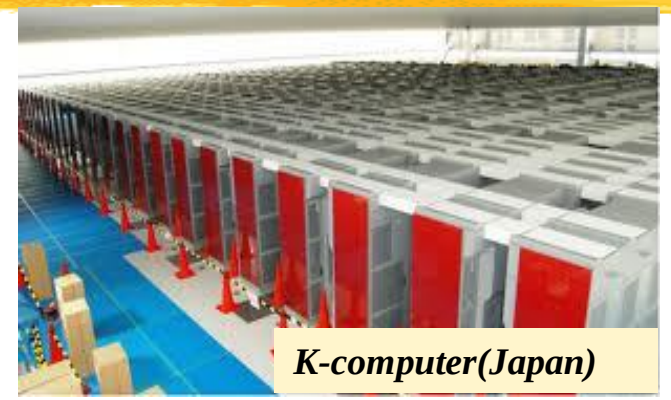
- Multiprocessor systems with more than one CPU in close communication.
- Improved Throughput, economical, increased reliability.
- Kinds:
  - Vector and pipelined
  - Symmetric and asymmetric multiprocessing
  - Distributed memory vs. shared memory
- Programming models:
  - Tightly coupled vs. loosely coupled ,message-based vs. shared variable

Parallel Computing Systems

ILLIAC 2 (UIllinois)



Climate modeling, earthquake simulations, genome analysis, protein folding, nuclear fusion research, …..



K-computer(Japan)

Connection Machine (MIT)

Tianhe-1(China)

IBM Blue Gene

Distributed Systems

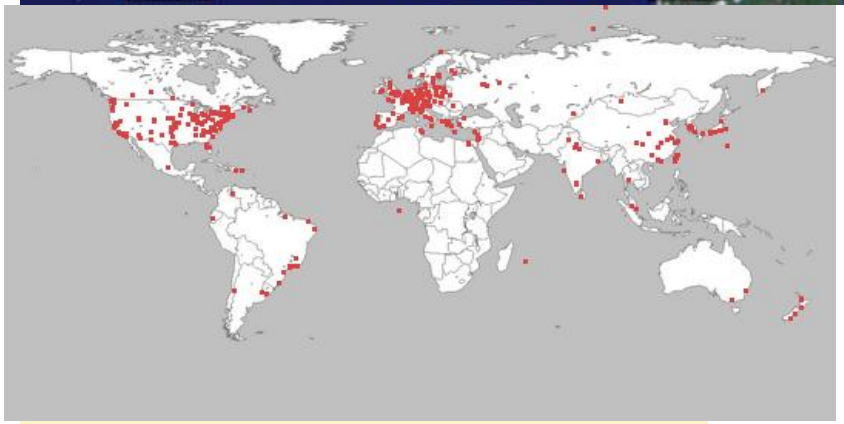
Hardware – *very cheap* ; Human – *very expensive*

- Distribute computation among many processors.
- Loosely coupled -
  - no shared memory, various communication lines
- client/server architectures
- Advantages:
  - resource sharing
  - computation speed-up
  - reliability
  - communication - e.g. email
- Applications - digital libraries, digital multimedia

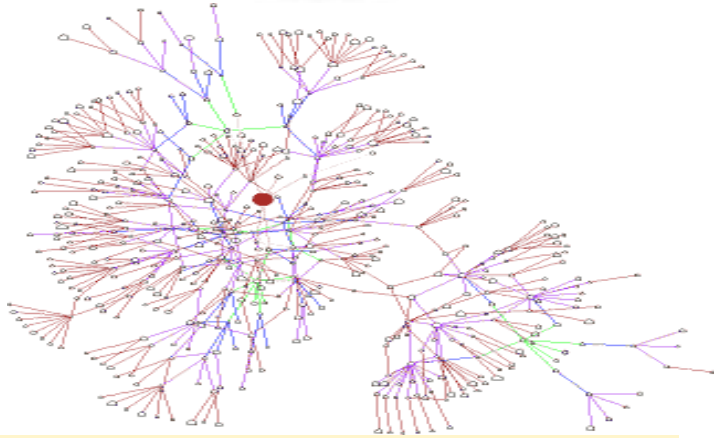
Distributed Computing Systems

Globus Grid Computing Toolkit

Cloud Computing Offerings



PlanetLab



Gnutella P2P Network

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too long.
  - Useful in applications such as multimedia, virtual reality.

Summary of lecture

- What is an operating system?
- Early Operating Systems
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems