EECS 262a
Advanced Topics in Computer Systems
Lecture 1

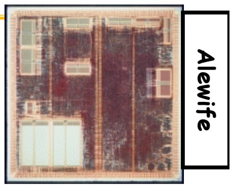
Introduction/UNIX
January 20th, 2016

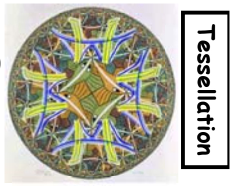
John Kubiatowicz
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~kubitron/cs262a

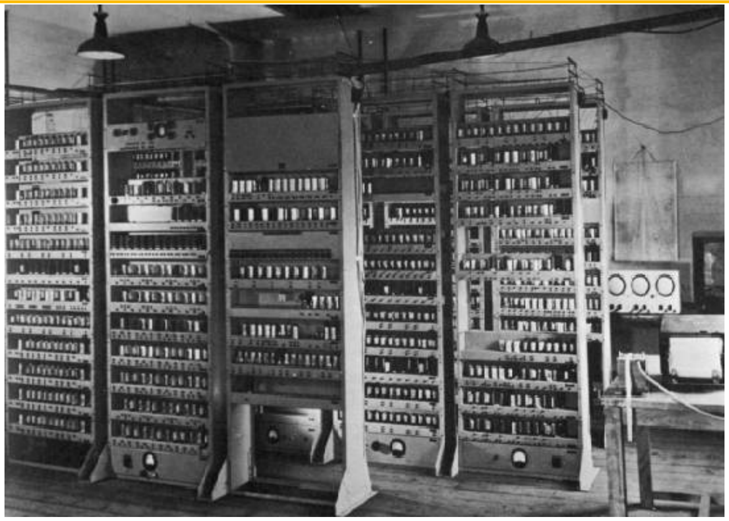
Who am I?

- **Professor John Kubiatowicz (Prof "Kubi")**
  - Background in Hardware Design
    - » Alewife project at MIT
    - » Designed CMMU, Modified SPAR C processor
    - » Helped to write operating system
  - Background in Operating Systems
    - » Worked for Project Athena (MIT)
    - » OS Developer (device drivers, network file systems)
    - » Worked on Clustered High-Availability systems
    - » OS lead researcher for the new Berkeley SwarmLab (SwarmOS). More later.
  - Peer-to-Peer
    - » OceanStore project – Store your data for 1000 years
    - » Tapestry and Bamboo – Find you data around globe
  - SwarmLAB
    - » Global Data Plane (GDP)
  - Quantum Computing
    - » Exploring architectures for quantum computers
    - » CAD tool set yields some interesting results





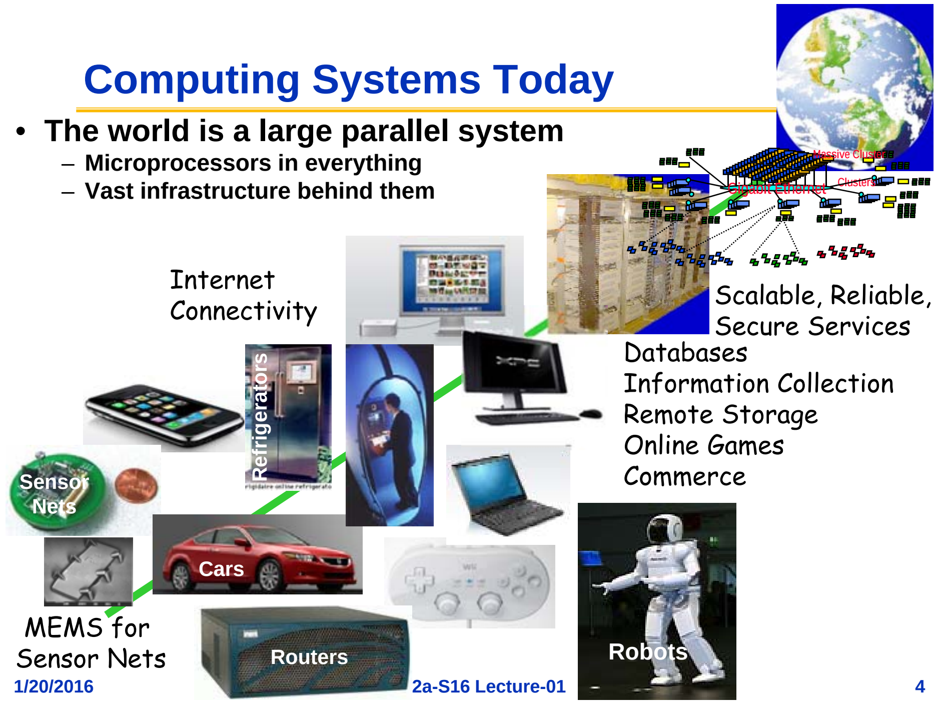
Computing Devices Then…



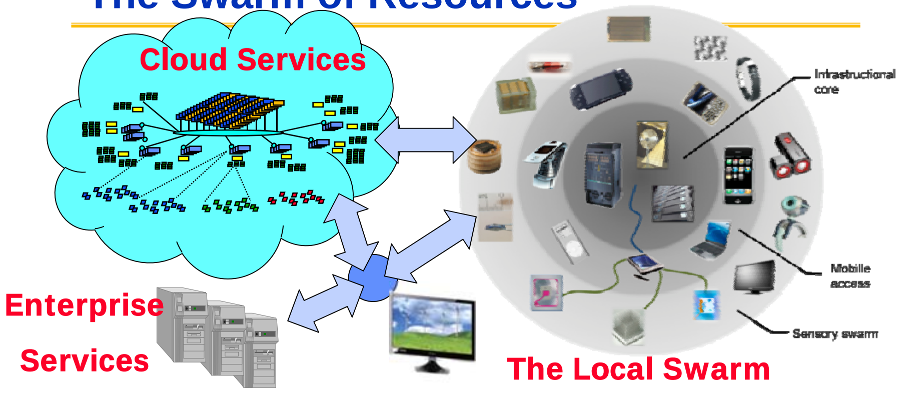
EDSAC, University of Cambridge, UK, 1949

Computing Systems Today

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

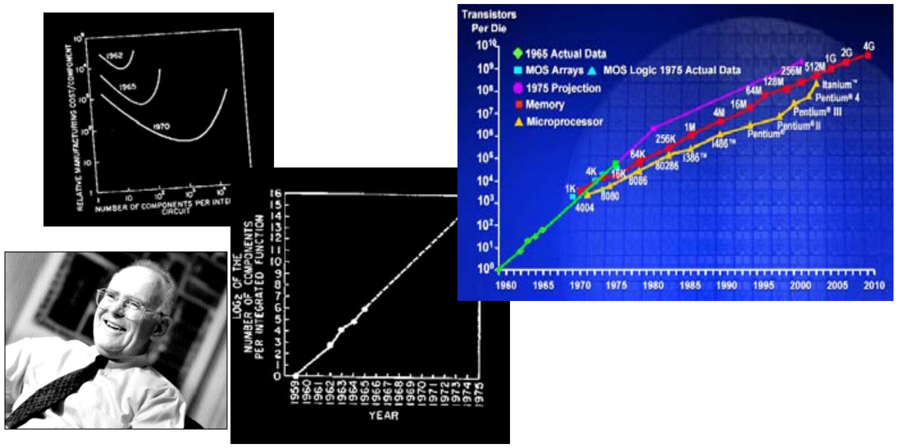


The Swarm of Resources



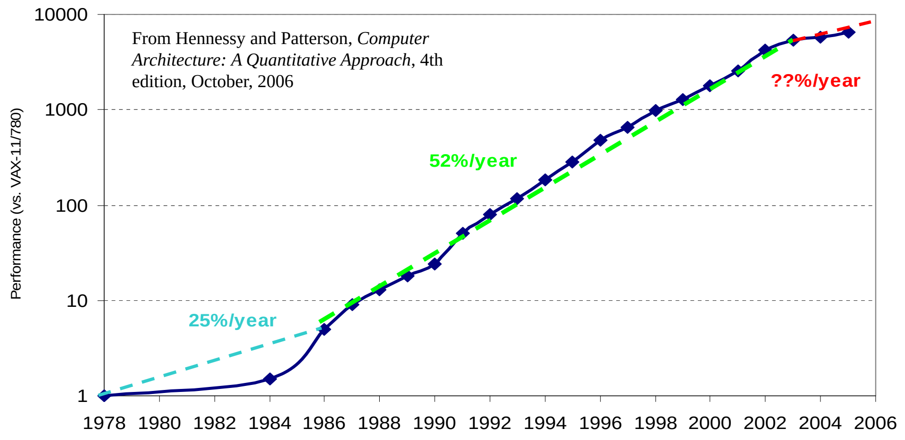
- What system structure required to support Swarm?
  - Discover and Manage resource
  - Integrate sensors, portable devices, cloud components
  - Guarantee responsiveness, real-time behavior, throughput
  - Self-adapting to adjust for failure and performance predictability
  - Uniformly secure, durable, available data

Moore's Law



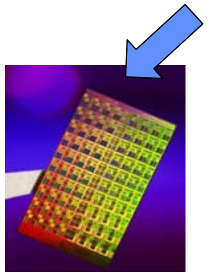
- "Cramming More Components onto Integrated Circuits"
  - Gordon Moore, Electronics, 1965
- # on transistors on cost-effective integrated circuit double every 18 months

Crossroads: Uniprocessor Performance

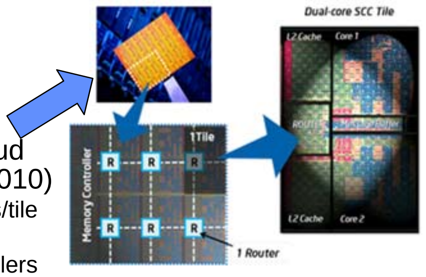


- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

ManyCore Chips: The future is here

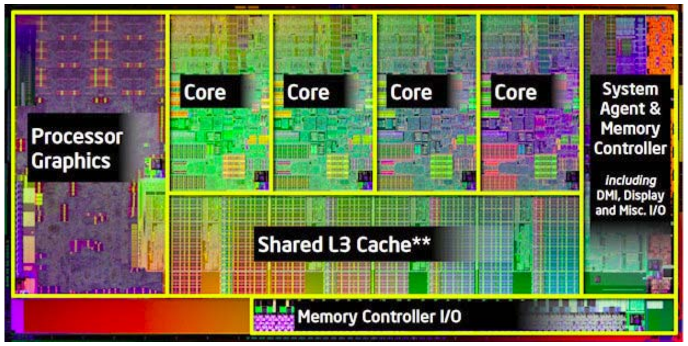


- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors



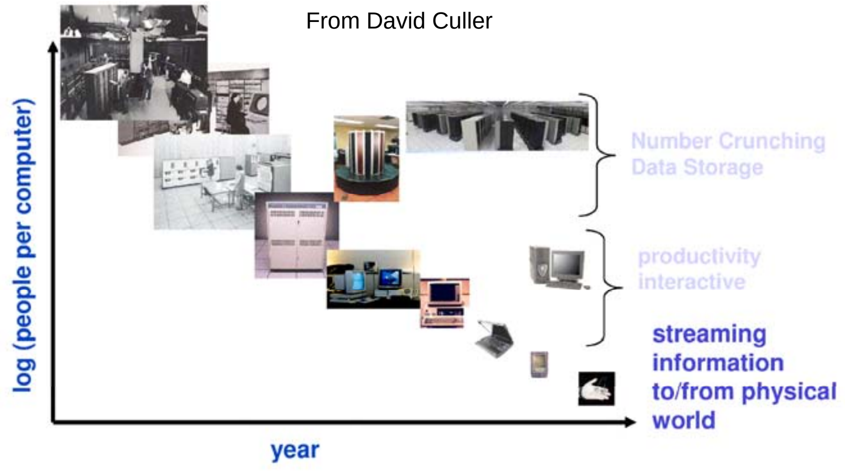
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers
  - Hardware support for message-passing
- "ManyCore" refers to many processors/chip
  - 64? 128? Hard to say exact boundary
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels

SandyBridge die photo: The world is parallel!



- Package: LGA 1155
  - 1155 pins
  - 95W design envelope
- Cache:
  - L1: 32K Inst, 32K Data (3 clock access)
  - L2: 256K (8 clock access)
  - Shared L3: 3MB – 20MB (not out yet)
- Transistor count:
  - 504 Million (2 cores, 3MB L3)
  - 2.27 Billion (8 cores, 20MB L3)
- Note that ring bus is on high metal layers – above the Shared L3 Cache

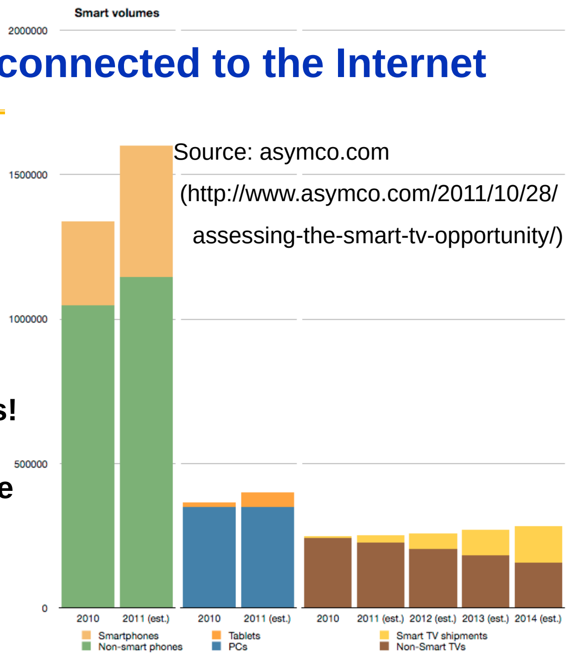
People-to-CPUs Ratio Over Time



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

Not Only PCs connected to the Internet

- 2011 shipments:
  - 454 million smartphones
  - 352 million PCs
  - 51 million tablets
  - 25 million smart TVs
- Smartphone shipments now exceed PC shipments!
- 4 billion phones in the world → smartphone over next decade

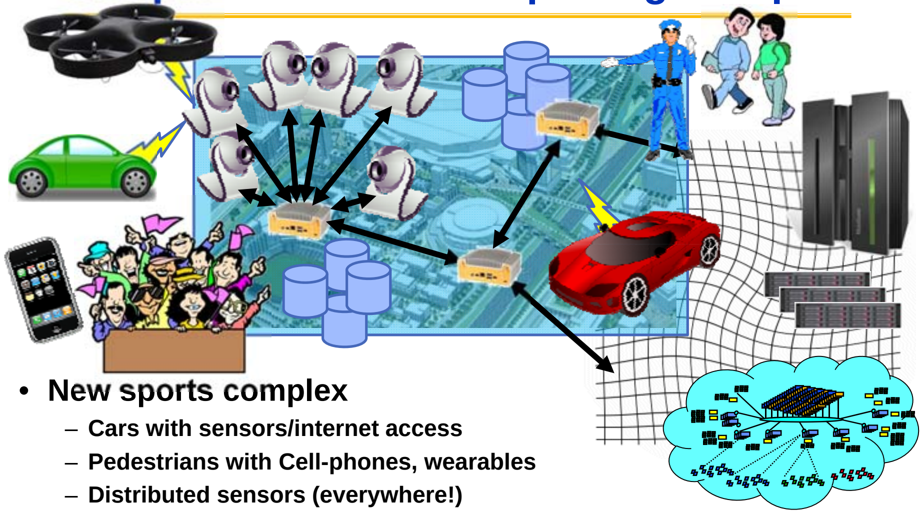


Source: asymco.com (http://www.asymco.com/2011/10/28/assessing-the-smart-tv-opportunity/)

Systems Challenges

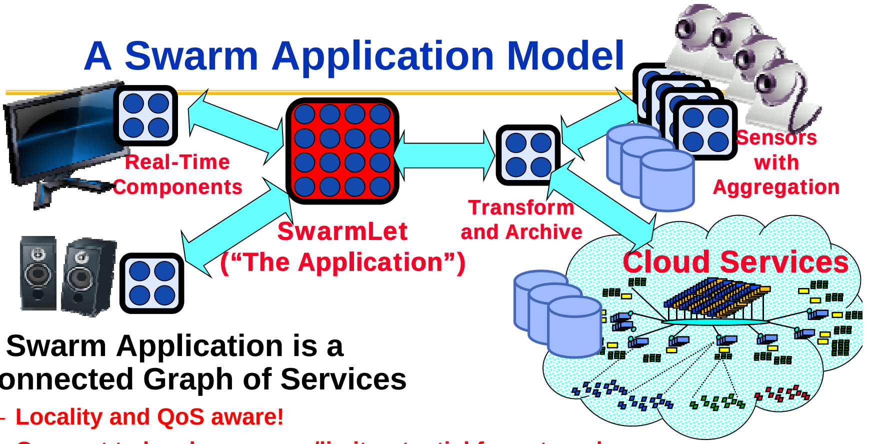
- Enormous scale, heterogeneity, and dynamic range:
  - CPU: sensor motes → GPUs
    - » Cores: one → 100s [2-orders of magnitude variation]
    - » Clusters: few machines → 10,000s machines [4 orders of mag.]
  - Network: Inter-core networks → Internet
    - » Latency: nanosecs → secs (satellite) [9 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

Sample Environment: Sporting Complex



- New sports complex
  - Cars with sensors/internet access
  - Pedestrians with Cell-phones, wearables
  - Distributed sensors (everywhere!)
  - Climate control
  - Computation: secure processing of sensitive information

A Swarm Application Model



- A Swarm Application is a Connected Graph of Services
  - Locality and QoS aware!
  - Connect to local resources/limit potential for external observability/interference
- Distributed storage everywhere
- Location-independent addressing of components
- Hardware and topology easy to postulate – but: How to make it easy to program???

Information-Centric Vision

- All that really matters is the information
  - Integrity, Privacy, Availability, Durability
  - Hardware is fungible, replaceable, …
- Hardware assisted model when Necessary
  - Prevent accidental information leakage
  - Low energy authentication/encryption
- Should be accessible from anywhere
- Should be able to exploit locality
  - Data routed to *wherever* it is needed
  - Data transformed anywhere (Mobile Swarmlets!)

Goals of CS262a

- Common foundation for OS and database research
  - (Lower) Will throw in some hardware mechanisms and architecture
  - (Higher) Will throw in some distributed systems research as well
- Why common course?
  - OS and DB communities historically separate, despite much overlap in goals and ideas
  - Independent vocabulary, largely separate "classic" papers that the other community does not read
  - Particularly bad for OS folks, because DB is "just an application"
- In theory, this is part of a two-course sequence
  - 262b hasn't been taught in a long time!
  - Fortunately, 262a satisfies software breadth by itself

What is Research?

- Research = Analysis and Synthesis together
  - Analysis: understanding others' work – both the good AND the bad
  - Synthesis: finding new ways to advance the state of the art
- Systems research isn't cut-and-dried
  - Few "provably correct" answers
  - In fact, some outright confusion/bad results
- Analysis in CS262a: Literature survey
  - Read, analyze, criticize papers
  - All research projects fail in some way
  - Successful projects: got some interesting results anyway
- Synthesis in CS262a: Do a small piece of real research
  - Suggested projects handed out in 3-4 weeks
  - Teams of 2 or 3
  - Poster session and "conference paper" at end of the semester
  - Usually best papers make it into a real conferences (with extra work)

Basic Format of CS262a classes

- Paper-Driven Lecture format with Discussion
  - Cover 1-2 papers plus announcements
  - Lots of discussion!!! Guest Lecturers!
  - In fact, prefer far more discussion, much less lecturing
- Will not cover too much basic material
  - We assume that you have had an introductory OS course
  - If not, you will have to work hardware to understand the material
  - Many online videos of CS162. For instance:
    - » http://www.infocobuild.com/education/audio-video-courses /computer-science/cs162-fall2010-berkeley.html
- Class preparation:
  - Reading papers is hard, especially at first
  - Every class has 1 or 2 papers that will be the focus of the class
  - Read BEFORE CLASS
  - If you need background, read background papers BEFORE CLASS

Basic Format of CS262a classes (con't)

- Homework: brief summary for every paper
  - ½ page or less/paper
  - 2 most important things in paper
  - 1 major flaw
- Summaries must be submitted through online submission form *before class* in order to get credit
  - You can skip 3 summaries without penalty
  - After that, they eat into your grade
- This is a grad class $\Rightarrow$ material may be controversial
  - (HOPEFULLY?)

Topics: All about sharing (reliably, securely!)

- Course topics (not necessarily in this order)
  - OS structure: File Systems, Virtual Memory ….
  - Transactions, recovery, & Fault-tolerance
  - Concurrency, scheduling, synchronization
  - Communication, Distributed Systems
  - Multiprocessors
  - Protection and Security
  - "Revealed Truth" – overall principles!
- Mix of Historical papers and Recent research
  - Will be updating the reading list from previous terms
  - Will also be reading some classic papers
- Why is History important?
  - All systems exist in a context. Historical papers part of culture!
  - Many ideas and techniques keep recurring (or never leave)
  - Need to avoid the "short horizon" effect
    - » Claim you invented something that originally appeared in 1975

Research Project

- Research-oriented course
  - Project provides opportunity to do “research in the small” to help make transition from good student to research colleague
  - Assumption is that you will advance the state of the art in some way
  - Projects done in groups of 2 or 3 students
- Topic?
  - Should be topical to CS262
  - Exciting possibilities related to SWARM-LAB, AMP-LAB
  - We will provide project suggestions in a couple of weeks
- Details:
  - meet multiple times with faculty to see progress
  - give poster session at end of semester
  - written report like conference paper
  - Usually, best papers make it into a real conference (with extra work)
- Can you share a project with other systems courses?
  - Under most circumstances, the answer is “yes”
  - Need to ok with me, however

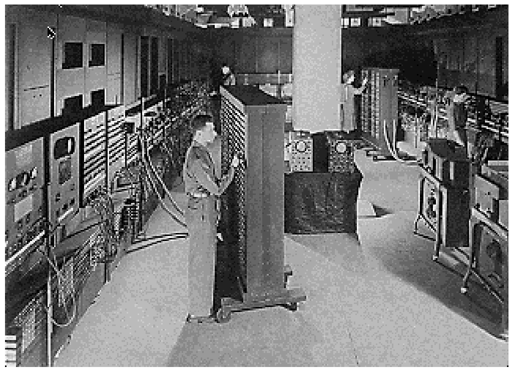
More Course Info

- Website:
  - http://www.cs.berkeley.edu/~kubitron/cs262
  - Contains lecture notes, readings, announcements
- Grading:
  - 45% Project Paper
  - 15% project demo
  - 30% Midterm
  - 10% Project summaries
- Signup:
  - Please go to the “Enrollment” link and sign up for the class
    - » Need this in order to submit summaries
- Summary details
  - Can miss 3 summaries without penalty, no reason needed (or wanted)
  - Summaries: < ½ per paper
    - » At least 1 criticism, 2 important insights from paper
    - » Online submission form

Coping with Size of Class

- Cannot get interaction with this many people
  - In past, have had an entrance exam
  - Instead, we will probably cap the class at 50 people (maximum overcapacity for this room).
- Graduate Students working on Prelim breadth requirement have precedence
- Undergraduates: only if there is room
  - We will take people only if they did well in CS162
  - We will take them in order of the wait list
- Plea: if not really interested in this class, let others take your slot!

Dawn of time ENIAC: (1945—1955)

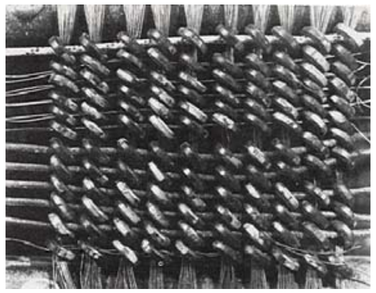


- “The machine designed by Drs. Eckert and Mauchly was a monstrosity. When it was finished, the ENIAC filled an entire room, weighed thirty tons, and consumed two hundred kilowatts of power.”
- http://ei.cs.vt.edu/~history/ENIAC.Richey.HTML

History Phase 1 (1948—1970) Hardware Expensive, Humans Cheap

- When computers cost millions of $'s, optimize for more efficient use of the hardware!
  - Lack of interaction between user and computer
- User at console: one user at a time
- Batch monitor: load program, run, print
- Optimize to better use hardware
  - When user thinking at console, computer idle⇒BAD!
  - Feed computer batches and make users wait
  - Autograder for this course is similar
- *No protection:* what if batch program has bug?

Core Memories (1950s & 60s)


The first magnetic core memory, from the IBM 405 Alphabetical Accounting Machine.

- Core Memory stored data as magnetization in iron rings
  - Iron "cores" woven into a 2-dimensional mesh of wires
  - Origin of the term "Dump Core"
  - Rumor that IBM consulted Life Saver company
- See: http://www.columbia.edu/acis/history/core.html

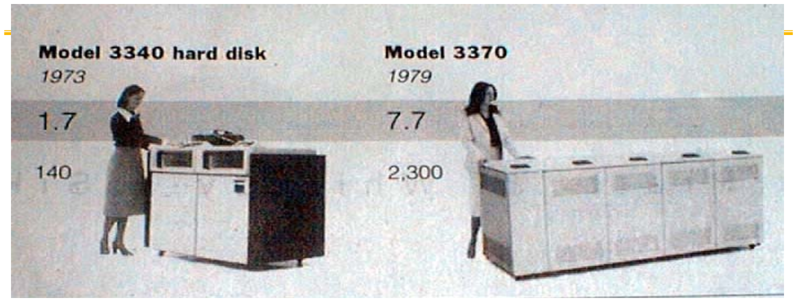
History Phase 1½ (late 60s/early 70s)

- Data channels, Interrupts: overlap I/O and compute
  - DMA – Direct Memory Access for I/O devices
  - I/O can be completed asynchronously
- Multiprogramming: several programs run simultaneously
  - Small jobs not delayed by large jobs
  - More overlap between I/O and CPU
  - Need memory protection between programs and/or OS
- Complexity gets out of hand:
  - Multics: announced in 1963, ran in 1969
    - » 1777 people "contributed to Multics" (30-40 core dev)
    - » Turing award lecture from Fernando Corbató (key researcher): "On building systems that will fail"
  - OS 360: released with 1000 known bugs (APARs)
    - » "Anomalous Program Activity Report"
- OS finally becomes an important science:
  - How to deal with complexity???
  - UNIX based on Multics, but vastly simplified

A Multics System (Circa 1976)

- The 6180 at MIT IPC, skin doors open, circa 1976:
  - "We usually ran the machine with doors open so the operators could see the AQ register display, which gave you an idea of the machine load, and for convenient access to the EXECUTE button, which the operator would push to enter BOS if the machine crashed."
- http://www.multicians.org/multics-stories.html

Early Disk History

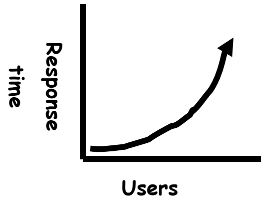


1973:	1979:
1. 7 Mbit/sq. in	7. 7 Mbit/sq. in
140 MBytes	2,300 MBytes

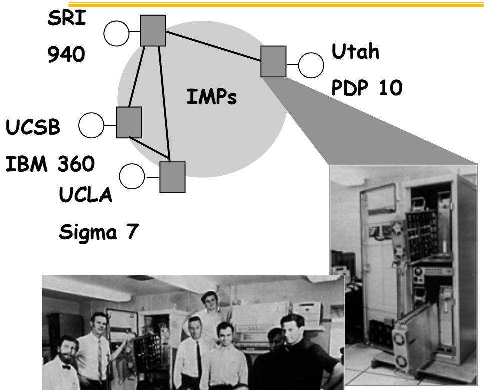
Space: 8TB in 3½ inch form factor!
(Seagate, Nov 2014)
Areal Density: over 1TB/square inch (SMR)

History Phase 2 (1970 – 1985) Hardware Cheaper, Humans Expensive

- Computers available for tens of thousands of dollars instead of millions
- OS Technology maturing/stabilizing
- *Interactive* timesharing:
  - Use cheap terminals (~$1000) to let multiple users interact with the system at the same time
  - Sacrifice CPU time to get better response time
  - Users do debugging, editing, and email online
- Problem: Thrashing
  - Performance very non-linear response with load
  - Thrashing caused by many factors including
    - » Swapping, queueing



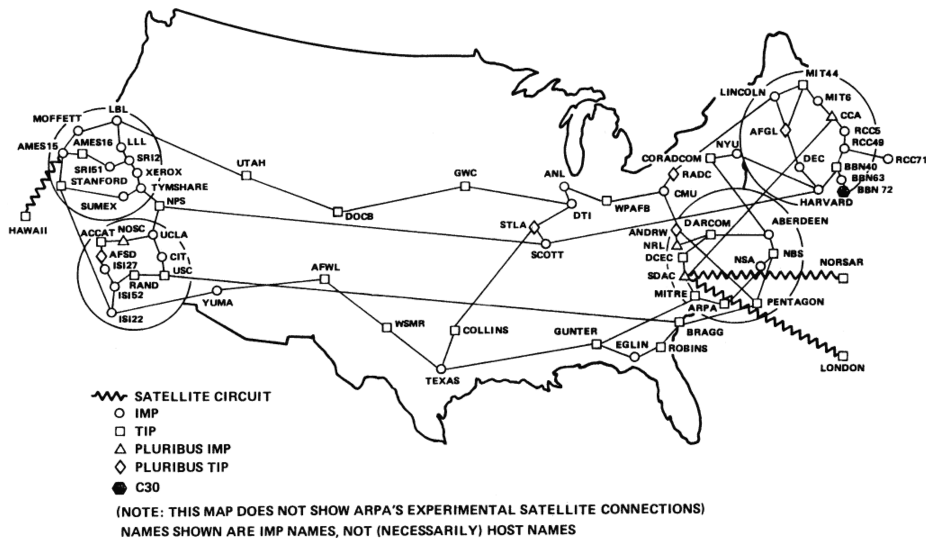
The ARPANet (1968-1970's)



- Paul Baran
  - RAND Corp, early 1960s
  - Communications networks that would survive a major enemy attack
- ARPANet: Research vehicle for "Resource Sharing Computer Networks"
  - 2 September 1969: UCLA first node on the ARPANet
  - December 1969: 4 nodes connected by 56 kbps phone lines
  - 1971: First Email
  - 1970's: <100 computers

BBN team that implemented
the interface message processor (IMP)

ARPANET GEOGRAPHIC MAP, OCTOBER 1980



(NOTE: THIS MAP DOES NOT SHOW ARPA'S EXPERIMENTAL SATELLITE CONNECTIONS)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

History Phase 3 (1981—)
Hardware Very Cheap, Humans Very Expensive

- Computer costs $1K, Programmer costs $100K/year
  - If you can make someone 1% more efficient by giving them a computer, it's worth it!
  - Use computers to make people more efficient
- Personal computing:
  - Computers cheap, so give everyone a PC
- Limited Hardware Resources Initially:
  - OS becomes a subroutine library
  - One application at a time (MSDOS, CP/M, …)
- Eventually PCs become powerful:
  - OS regains all the complexity of a "big" OS
  - multiprogramming, memory protection, etc (NT,OS/2)
- Question: As hardware gets cheaper does need for OS go away?

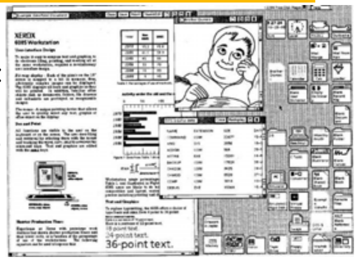
History Phase 3 (con't)
Graphical User Interfaces

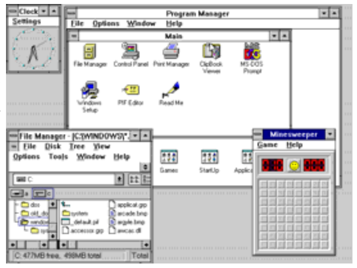
- Xerox Star: 1981
  - Originally a research project (Alto)
  - First "mice", "windows"
- Apple Lisa/Machintosh: 1984
  - "Look and Feel" suit 1988
  - Apple did it again and won: 2012
- Microsoft Windows:
  - Win 1.0 (1985)
  - Win 3.1 (1990)
  - Win 95 (1995)
  - Win NT (1993)
  - Win 2000 (2000)
  - Win XP (2001)
  - Win Vista (2007)
  - Win 7 (2009)

$\left.\begin{matrix}\text{Win 1.0 (1985)}\\ \text{Win 3.1 (1990)}\\ \text{Win 95 (1995)}\end{matrix}\right\}$ Single Level

$\left.\text{Win NT (1993)}\right\}$ HAL/Protection

$\left.\begin{matrix}\text{Win 2000 (2000)}\\ \text{Win XP (2001)}\\ \text{Win Vista (2007)}\\ \text{Win 7 (2009)}\end{matrix}\right\}$ No HAL/ Full Prot


Xerox Star


Windows 3.1

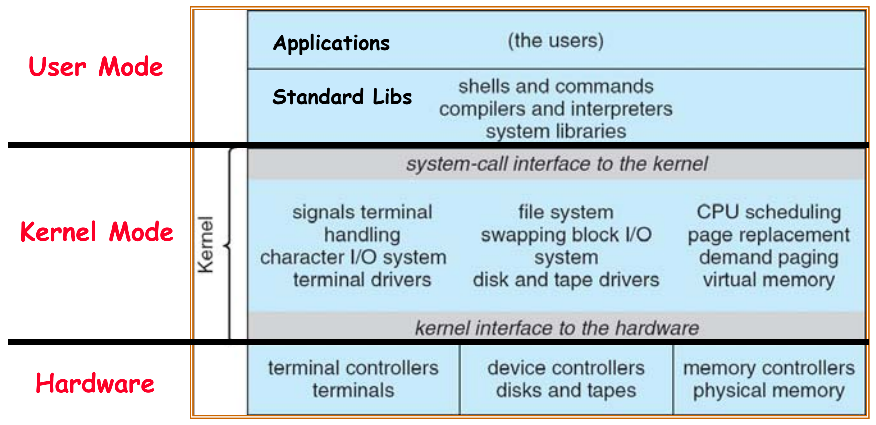
What about next phases of history?

- Let's save that for later
- On to UNIX Time sharing paper

The UNIX Time-Sharing System

- "Warm-up paper"
  - Date: July 1974
- Features:
  - Time-Sharing System
  - Hierarchical File System
  - Device-Independent I/O
  - Shell-based, tty user interface
  - Filter-based, Record-less processing Paradigm
- Version 3 Unix:
  - Ran on PDP-11s
  - < 50 KB
  - 2 man-years to write
  - Written in C
- Compare to Multics
  - 1777 people "contributed to Multics" (30-40 core dev)

UNIX System Structure



File System: The center of UNIX world

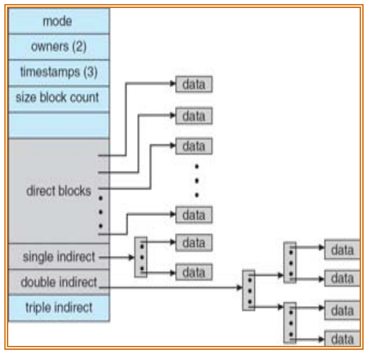
- Types of files:
  - Ordinary files (uninterpreted)
  - Directories (protected ordinary files)
  - Special files (I/O)
- Directories:
  - Root directory
  - Path Names
    - » Rooted Tree
    - » Current Working Directory mechanism
    - » Back link to parent directory
  - Multiple links to ordinary files
- Special Files
  - Uniform I/O model
  - Uniform naming and protection

File System Interface Details

- Removable file systems
  - Tree-structured
  - Mounted on an ordinary file
  - Appears in globally rooted tree after that
    - » Can have pathname for file that starts at root
- Protection:
  - User/World, RWX bits
  - Set-User-ID bit (SETUID)
  - Super User is special User-ID
- Uniform I/O model
  - Open, close, read, write, seek
  - Other system calls
  - Bytes, no records! No imposed semantics.
  - Long discussion about why no explicit locking mechanism

File System Implementation:

- Table of i-nodes
  - Short, unique name that points at file info
  - Allows simply and efficient file system repair (FSCK)
  - Can't handle accounting issues
  - Max file size: 2^{20} = 1 MB!
- Implementation Details:
  - Buffered data
  - Write behind (Some data only in memory!
  - Path name scanning
- BSD 4.1 version of same idea:
  - First 10 pointers are to data blocks
  - Ptr 11 points to "indirect block" containing 256 block ptrs
  - Ptr 12 points to "doubly indirect block" containing 256 indirect block ptrs for total of 64K blocks
  - Ptr13 points to a "triply indirect block" (16M blocks)



Other Ideas

- Processes and Images
  - Process consists of Text, Data, and Stack Segments
  - Process swapping
  - Essential mechanism:
    - » Fork() to create new process
    - » Wait() to wait for child to finish and get result
    - » Exit(Status) to exit and give status to parent
    - » Pipes for interprocess communication
    - » Exec(file, arg1, arg2, … argn) to execute new image
- Shell
  - Command processor: cmd arg1 ... argn
  - Standard I/O and I/O redirection
  - Multi-tasking from a single shell
  - Shell is just a program!
- Traps
  - Hardware Interrupts, Software Signals, Trap to system routine

Is this a good paper?

- What were the authors' goals?
- What about the performance metrics?
- Did they convince you that this was a good system?
- Were there any red-flags?
- What mistakes did they make?
- Does the system meet the "Test of Time" challenge?
- How would you review this paper today?