THE UNIVERSITY of EDINBURGH
informatics

Operating Systems (INFR10079)

2020/2021 Semester 2

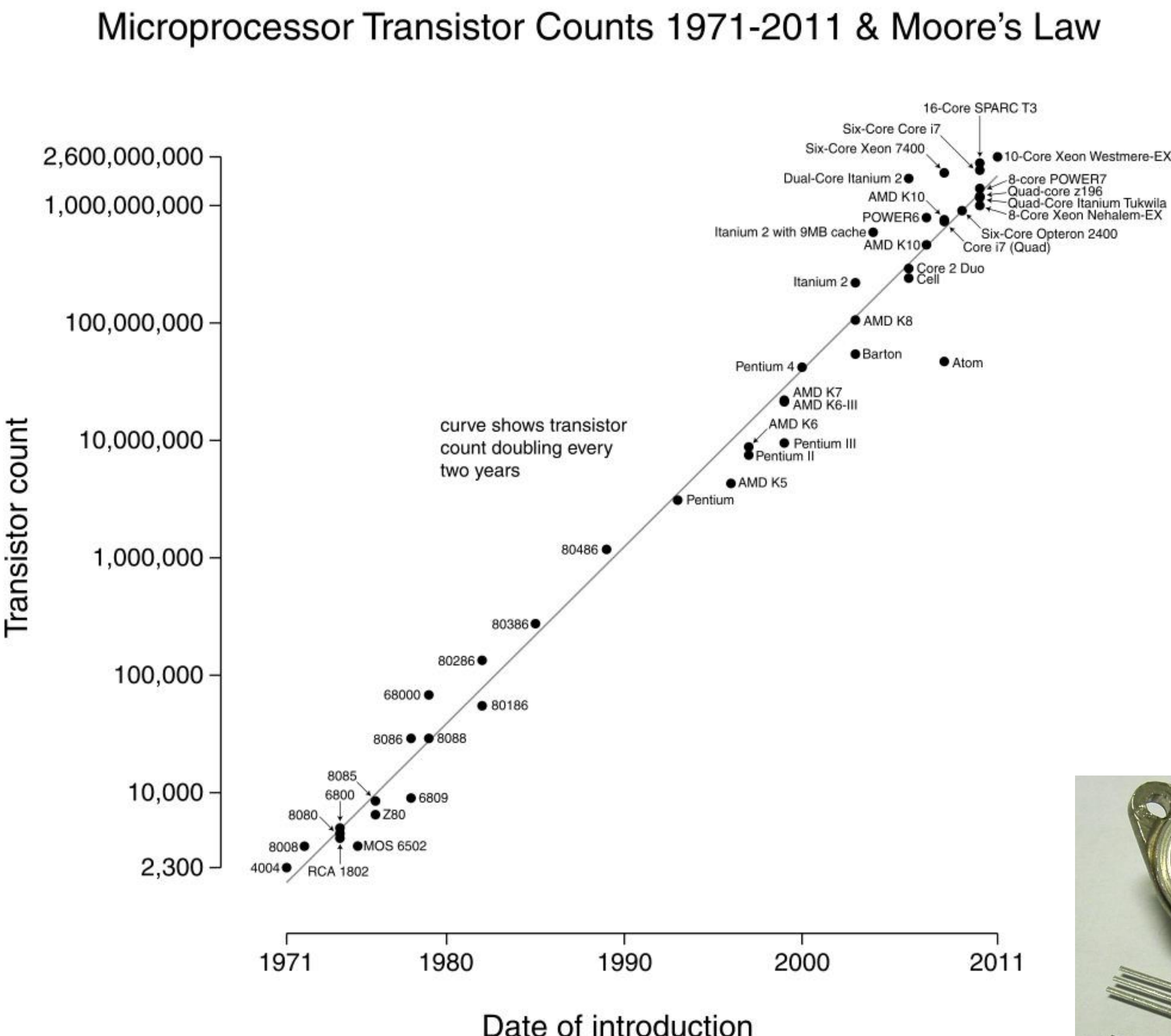
Introduction

(Operating Systems and Computing Systems History)

abarbala@inf.ed.ac.uk

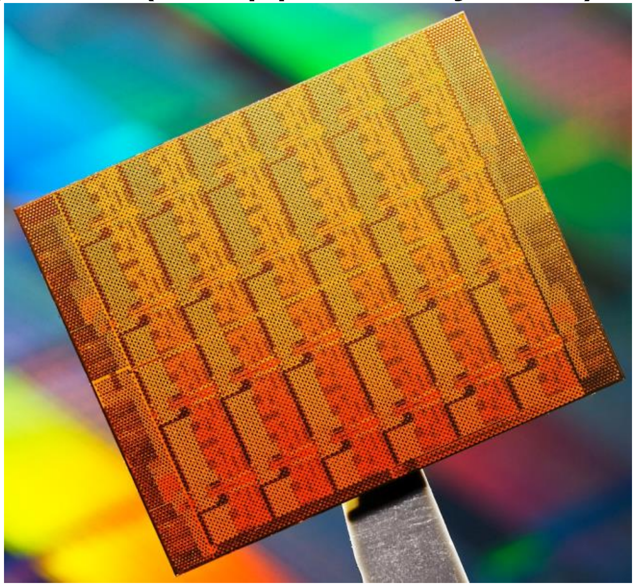
Chapter 1.4.1, 1.5, 1.6, 1.7, 1.8, 1.10, Appendix A

Hardware Complexity Increases



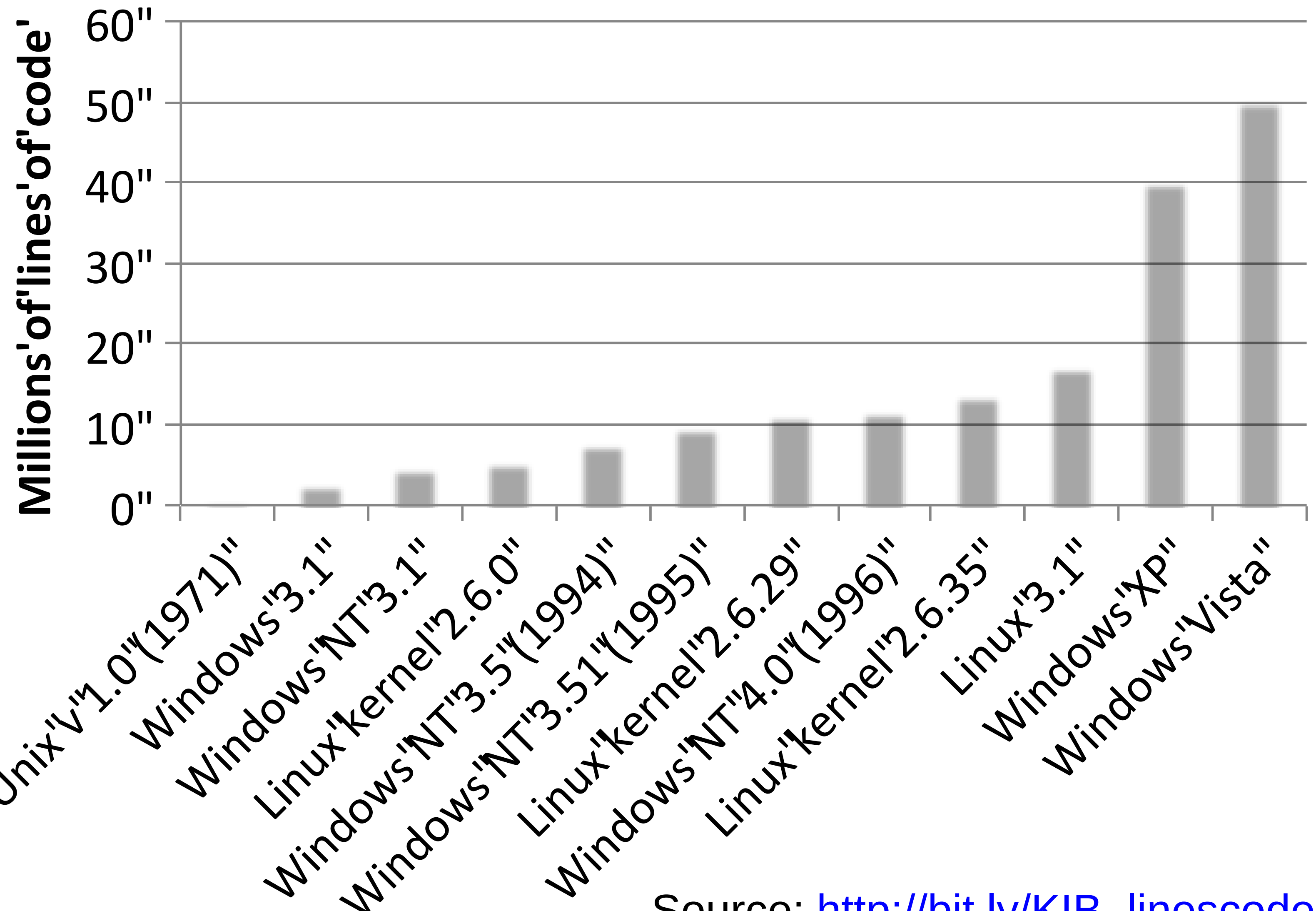
https://en.wikipedia.org/wiki/Transistor_count
https://en.wikipedia.org/wiki/Moore%27s_law

Moore's Law: 2X transistors/Chip Every 1.5 years (or approx. 2 years)



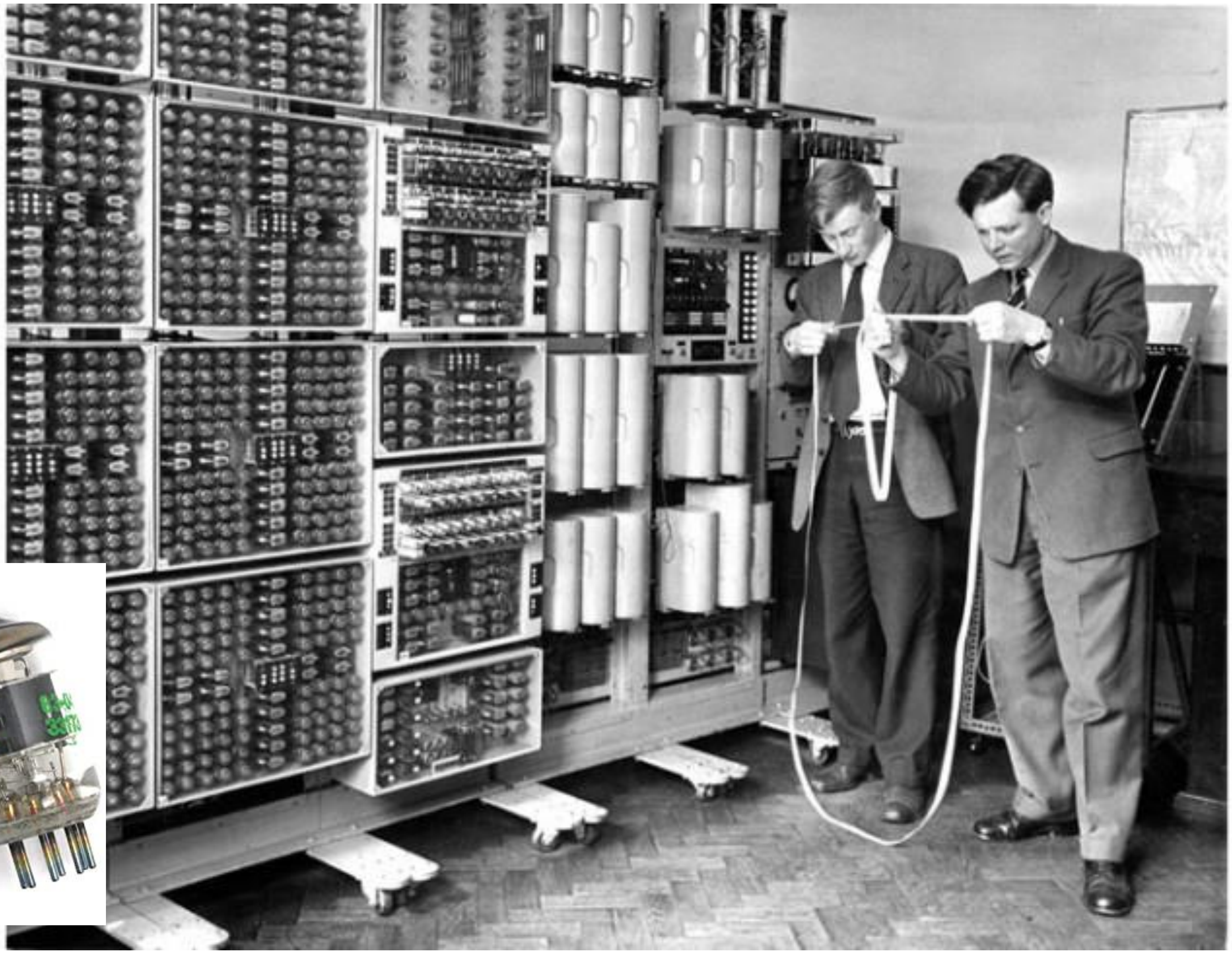
Transistor: the basic switching (0/1) component (discrete)

Software Complexity Increases



Source: http://bit.ly/KIB_linescode

Wolverhampton Instrument for Teaching Computing from Harwell (WITCH)



1951
No OS!

Run a specific calculation

https://www.extremetech.com/extreme/140953-worlds-oldest-original-digital-computer-is-turned-back-on-after-61-years

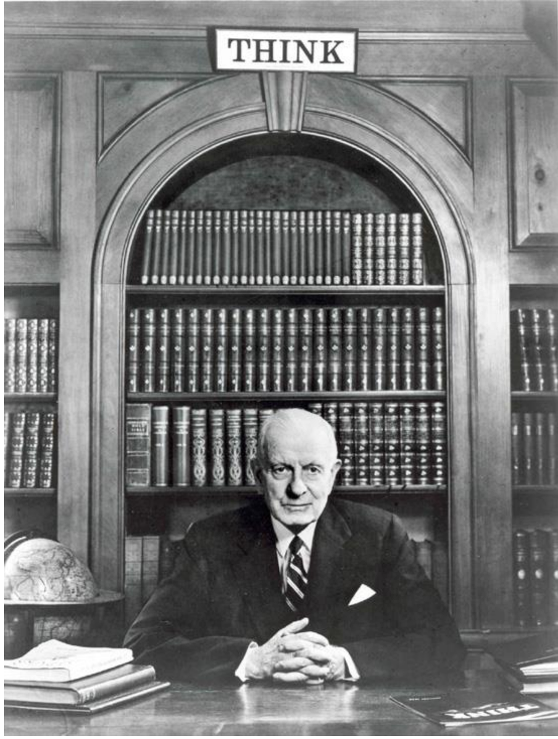
Hardware/Software Changes with Time

- **1960s:** mainframe computers (IBM)
- **1970s:** minicomputers (DEC)
- **1980s:** microprocessors and workstations (SUN), local-area networking, the Internet
- **1990s:** PCs (rise of Microsoft, Intel, Dell), the Web
- **2000s:**
  - Internet Services / Clusters (Amazon)
  - General Cloud Computing (Google, Amazon, Microsoft)
  - Mobile/ubiquitous/embedded computing (iPod, iPhone, iPad, Android)
- **2010s:** sensor networks, “data-intensive computing,” computers and the physical world
- **2020s:** it’s up to you!!

An OS History Lesson

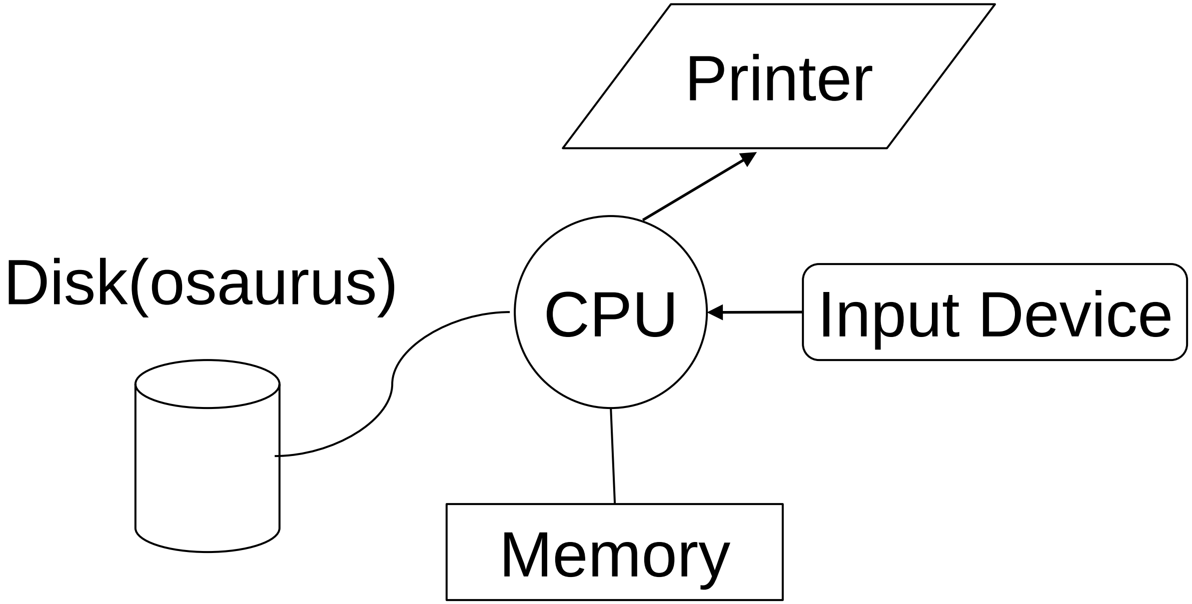
- Operating systems are the result of a **60 year long evolutionary process**
  - They were born out of need
- Learn about their evolution
  - Makes clear what some of their functions are, and why

At the Beginning...



- 1943
  - T.J. Watson (created IBM):
    " I think there is a world market for maybe <u>five computers</u>."
- Fast forward … 1950
  - There are maybe <u>20 computers in the world</u>
    - They were unbelievably expensive
    - Imagine this: machine time is more valuable than person time!
    - Ergo: efficient use of the hardware is paramount
  - Operating systems are born
    - They carry with them the vestiges of these ancient forces

Primordial Computer

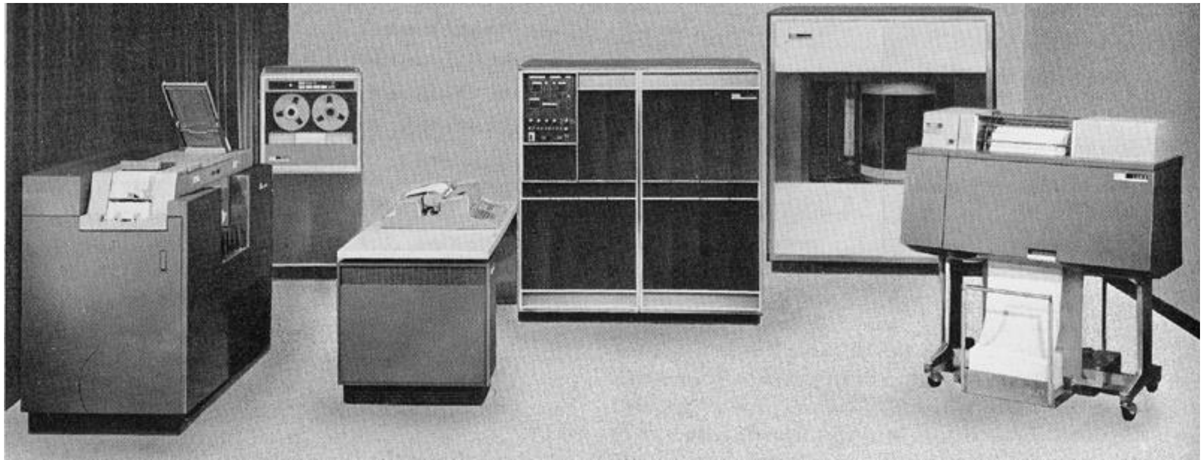


The OS as a Linked Library

- At the very beginning…
  - OS was just a library of code that you linked into your program
  - Programs were loaded in their entirety into memory, and executed
    - “OS” had an “API” that let you control
      - Disk
      - Printer
      - …
  - Interfaces were literally switches and blinking lights
  - When done running program, leave and turn the computer over to next person

Asynchronous I/O

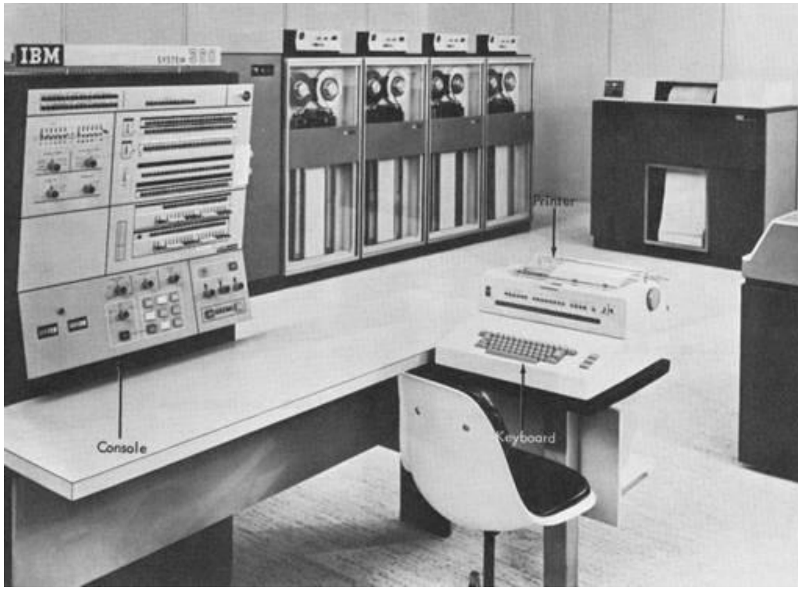
- The disk(osaurus) was really slow
- Add hardware so that the disk could operate without tying up the CPU
  - Disk controller
- Hot shot programmers could now write code that
  - Starts an I/O
  - Goes off and does some computing
  - Checks if the I/O is done at some later time
- Upside
  - Helps increase (expensive) CPU utilization
- Downsides
  - It's hard to get right
  - The benefits are job specific



IBM 1401

Multiprogramming

- To further increase system utilization: multiprogramming OSs
  - Multiple runnable jobs in memory at once
  - Overlaps I/O of one job with computing of another
    - While one job waits for I/O completion, another job uses the CPU
  - Can get rid of asynchronous I/O within individual jobs
    - Life of application programmer becomes simpler
    - Only the OS programmer needs to deal with asynchronous events
  - How to tell when devices are done?
    - Interrupts
    - Polling



IBM System 360

Timesharing

- To support interactive use, create a timesharing OS
  - multiple terminals for one machine
  - each user has illusion of entire machine to him/herself
  - optimize response time, perhaps at the cost of throughput
- Timeslicing
  - divide CPU equally among the users
  - if job is truly interactive (e.g., editor), then can jump between programs and users faster than users can generate load
  - permits users to interactively view, edit, debug running programs

- MIT CTSS system (operational 1961) among the first timesharing systems
  - only one user memory-resident at a time (32KB memory!)
- MIT Multics system (operational 1968) first large timeshared system
  - nearly all OS concepts can be traced back to Multics!
  - “second system syndrome”

Multiple Terminals

- In early 1980s, a single timeshared VAX-11/780 ran computing for all of CSE
- A typical VAX-11/780 was 1 MIPS (1 MHz) and had 1MB of RAM and 100MB of disk
- An Apple iPhone 4 is 1GHz (x1000), has 512MB of RAM (x512), and 32GB of flash (x320)

http://www.cs.man.ac.uk/CCS/res/res49.htm

Parallel Systems

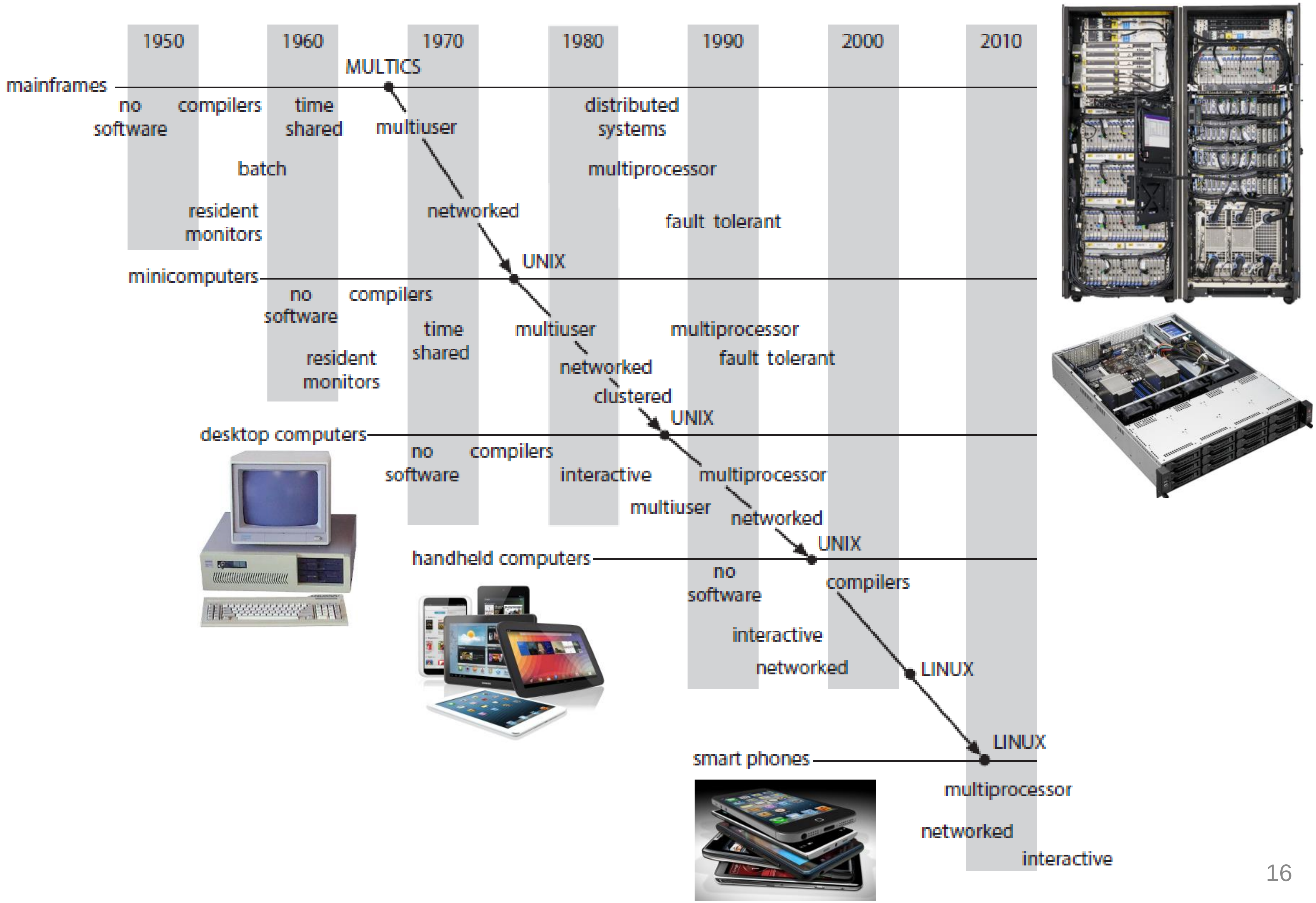
- Some applications can be written as multiple parallel threads or processes
  - can speed up the execution by running multiple threads/processes simultaneously on multiple CPUs [Burroughs D825, 1962]
  - need OS and language primitives for dividing program into multiple parallel activities
  - need OS primitives for fast communication among activities
    - degree of speedup dictated by communication/computation ratio
  - many flavors of parallel computers today
    - SMPs (symmetric multi-processors)
    - MPPs (massively parallel processors)
    - NOWs (networks of workstations)
    - Massive clusters (Google, Amazon.com, Microsoft)
    - Computational grid (SETI @home)

Personal Computing

- Primary goal was to enable new kinds of applications
- Bit mapped display [Xerox Alto,1973]
  - new classes of applications
  - new input device (the mouse)
- Move computing near the display
  - why?
- Window systems
  - the display as a managed resource
- Local area networks (Ethernet)
  - why?

Single Computer

Progression of Concepts and Form Factors



When a Single Computer is Not Enough

- Interconnect multiple computers together
- Supercomputers, clusters, data-center …

Distributed OS

- Distributed systems to facilitate use of geographically distributed resources
  - workstations on a LAN
  - servers across the Internet
- Supports applications running among multiple computers
  - interprocess communication
    - message passing, shared memory
  - networking stacks
- Sharing of distributed resources (hardware, software)
  - load balancing, authentication and access control, …

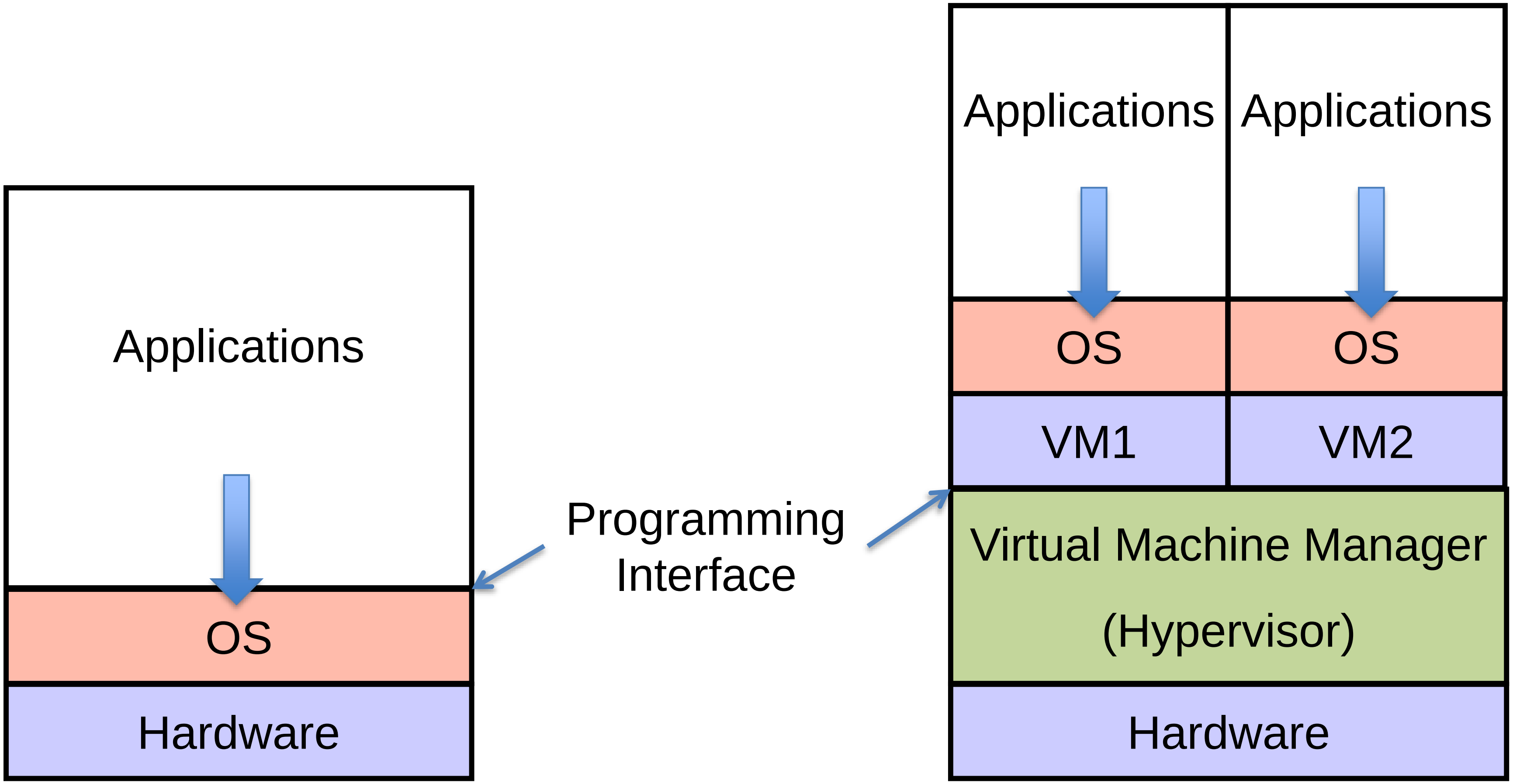
Client/Server Computing

- Mail server/service
- File server/service
- Print server/service
- Compute server/service
- Game server/service
- Music server/service
- Web server/service
- etc.

Peer-to-Peer (p2p) Systems

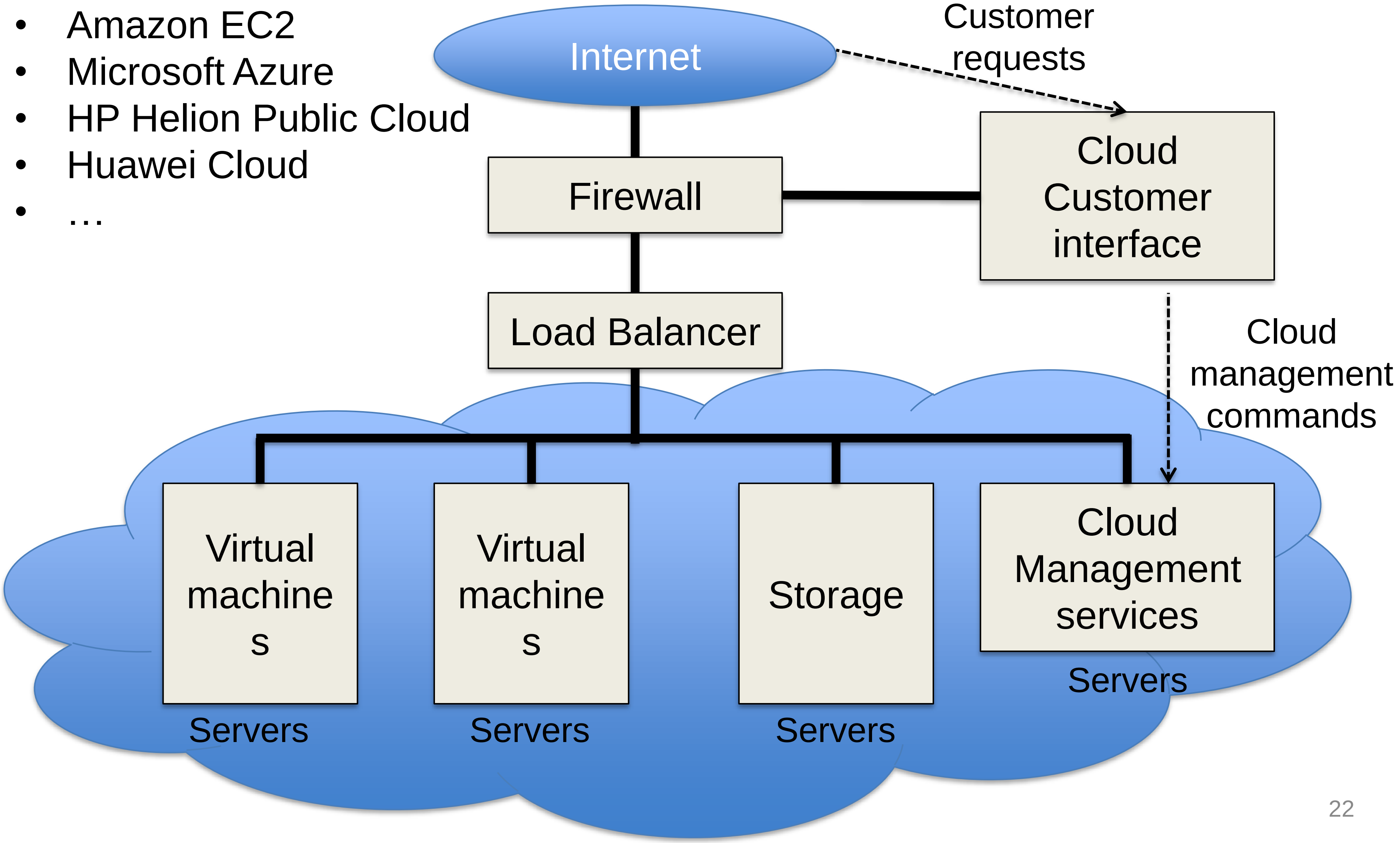
- Napster
- Gnutella
- BitTorrent
  - example technical challenge: self-organizing overlay network
  - technical advantage of BitTorrent?
  - er … legal advantage of BitTorrent?

Virtualization



Cloud Computing

- Amazon EC2
- Microsoft Azure
- HP Helion Public Cloud
- Huawei Cloud
- …



The major OS issues

- **Structure**: how is the OS organized?
- **Sharing**: how are resources shared across users?
- **Naming**: how are resources named (by users or programs)?
- **Security**: how is the integrity of the OS and its resources ensured?
- **Protection**: how is one user/program protected from another?
- **Performance**: how do we make it all go fast?
- **Reliability**: what happens if something goes wrong (either with hardware or with a program)?
- **Extensibility**: can we add new features?
- **Communication**: how do programs exchange information, including across a network?

More OS issues…

- **Concurrency**: how are parallel activities (computation and I/O) created and controlled?
- **Scale**: what happens as demands or resources increase?
- **Persistence**: how do you make data last longer than program executions?
- **Distribution**: how do multiple computers interact with each other?
- **Accounting**: how do we keep track of resource usage, and perhaps charge for it?

There are tradeoffs, not right and wrong!

Why Should One Learn Operating Systems?

- You may not ever build an OS
- But almost all code runs on top an OS
  - Almost every computing device runs an OS
- Hence, you need to understand the foundations
  - As a Computer Scientist or Computer Engineer
- Knowledge of how OSes work is crucial to proper, efficient, effective, and secure programming

It is a Great Time to Study Operating Systems!

- Many open-source OS projects
  - Check, study, and modify the code
  - Help find and fix bugs
- Emulators to run and debug OSes
  - Existent OSes
  - Historical/Future OSes (hardware not available)
- Renewed interest in OSes from major IT Companies
  - Rise of unikernels
  - Fully formally verified OSes
  - New OSes (e.g. Google Fucsia)