SE350: Operating Systems

Lecture 1: Introduction

Outline

- How do things work in SE350?
  - Instructional team
  - Course mechanics
  - Lab project
- What is an operating system?
- What makes designing operating systems challenging?

Teaching Team

Instructor: Seyed Majid Zahedi
zahedi@uwaterloo.ca
Office hours: M 2-3 and F 3:30-4:30 (DC-2524)

Lab instructor: Irene Huang
yqhuang@uwaterloo.ca
Office hours: TBA

SE350 TAs

Ali Hossein Abbasi Abyaneh
a36hosse@uwaterloo.ca

Huanyi Chen
h365chen@uwaterloo.ca

Weitian Xing
w23xing@uwaterloo.ca

Online Resources

- Course website 

https://ece.uwaterloo.ca/~smzahedi/crs/se350/

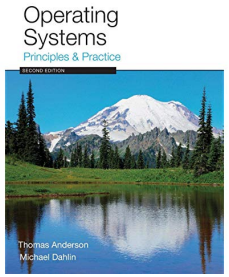
- Course on Piazza 

https://piazza.com/uwaterloo.ca/winter2020/se350

Readings

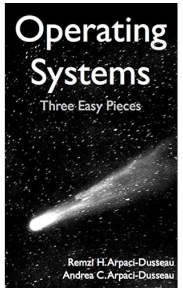
- Main textbook

  Operating Systems: Principles and Practice (2nd Edition)

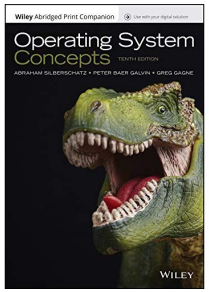


- Optional references

  Operating Systems: Three Easy Pieces (Freely Available)



  Operating System Concepts (10th Edition)



Evaluation

- Midterm exam: 15%
  - Feb 27th, 18:30-20:20 (DC-1351)
- Lab project: 35%
- Final exam: 50%

RTX Project Overview

- What is this project about?
- What are deliverables?
- How to form and leave project groups?
- What lab facilities are provided?
- How to seek help outside lab?

RTX Project

- You will design, implement, and test real-time executive (RTX)
  - Basic multiprogramming environment
  - Five priority queues and preemption
  - Simple memory management
  - Message-based Inter-Process Communication (IPC)
  - System console I/O (UART 0/1)
  - Debugging support (UART 1/0)
- Your RTX operates on Keil MCB1700 Cortex-M3 boards

Four Milestones

- RTX implementations (P1, P2, and P3)
  - Optional: combine P2 and P3
  - Make your decision by 16:30 on March 13th
- RTX demonstrations (Weeks 5, 10, and 12)
  - Week 10 is help session if P2 and P3 are combined
- RTX P4 is final project report (30-40 pages) + code
- Three grace days without penalty
  - 10% per day late submission penalty afterwards
  - Submissions will not be accepted after three days
- Zero tolerance policy for plagiarism
  - We use Moss
  - We follow UW Policy 71 for any single incident

Project Groups

- Forming project groups
  - Four members (three is acceptable for special cases)
  - Within the same lab section as much as possible
  - Use LEARN to signup by 16:30 on Jan. 17th
- Leaving project groups
  - One week notice in writing before nearest deadline
  - Only one split-up is allowed
  - All students involved lose one grace day
- Everyone in the same group gets the same grade

Lab Facilities

- • Five scheduled lab sessions are in weeks 3,5, 8,10 and 12
  - M, W, and F (excluding Reading Week)
  - Help sessions: Weeks 3 and 8
  - Demo sessions: Weeks 5, 10 and 12
- Lab is in E2-2363
  - Nexus computers
  - Keil MCB1700 LPC1768 (Cortex-M3) boards
  - MDK-ARM MDK-Lite ed. V4.60 (32KB code size limit)
    - RealView Compilation Tools are included
    - Simulator is good for development work outside lab*
- SE350 shares lab with ECE222 sessions which are scheduled in even weeks (T, W, and Th)

* Simulator may not perfectly match hardware behaviour, so test your code on hardware well before deadlines!

Seeking Help Outside Lab

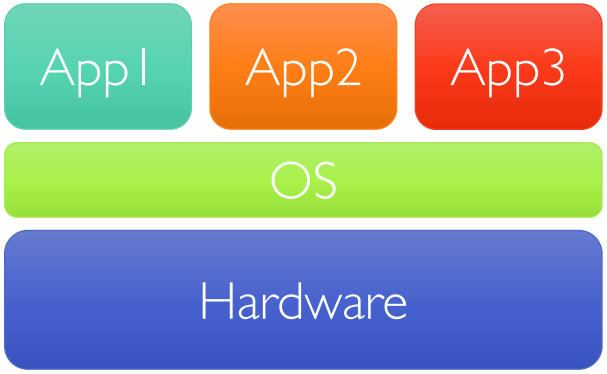
- Lab Q&A on Piazza discussion forum
  - Looking for group partners
  - Lab/Project administration
  - Keil IDE Q&A
  - Project Q&A
  - Target response time: one business day
  - Do not wait till the last minute to ask questions
- Individual emails
  - Only for questions containing confidential information
- Office hours during non-lab weeks
  - No office hours during Reading Week and Midterm week
- By appointment

Important Near-Term Task

Signup for project groups by 16:30 on Jan 17th

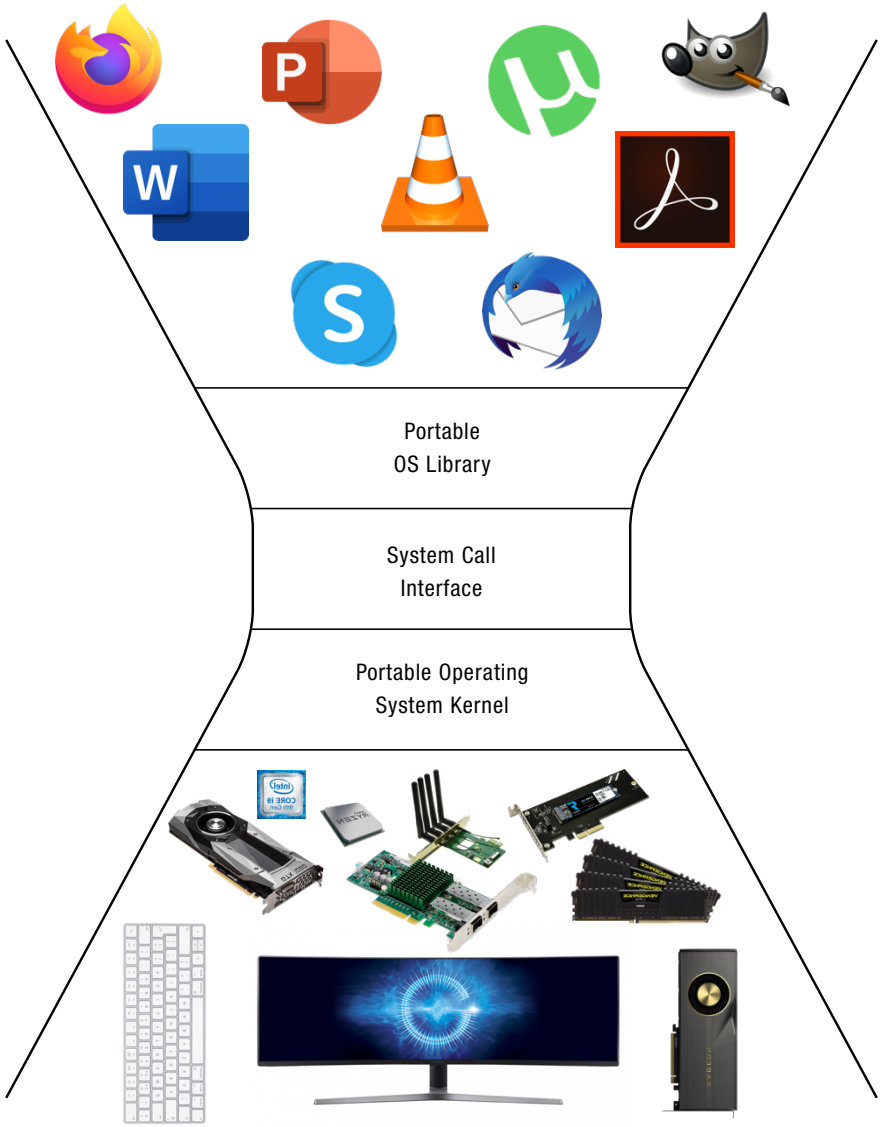
What is an Operating System?

- Special layer of software that provides applications access to hardware resources
  - Abstract view of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities

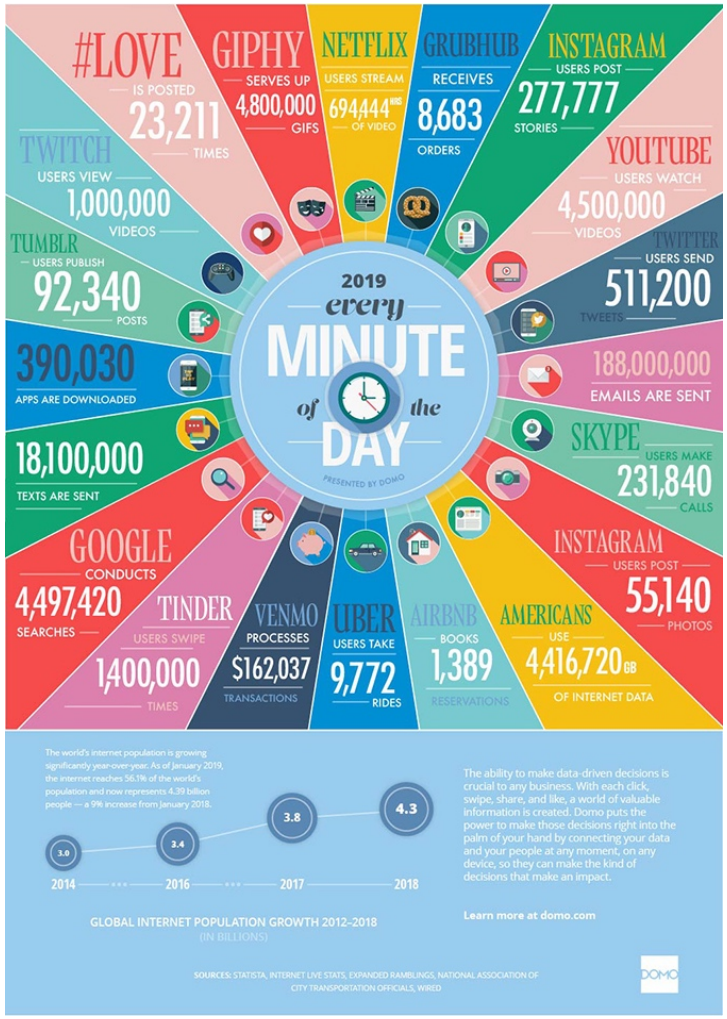


What Do Operating Systems Do?

- Provide abstractions to applications
  - File systems
  - Processes, threads
  - Virtual memory
  - Naming system, …
- Manage diverse resources
  - Memory, CPU, storage, …
- Achieves above by implementing specific algorithms and techniques
  - Scheduling
  - Concurrency
  - Transactions
  - Security, …



What Makes Designing OSs Exciting and Challenging?

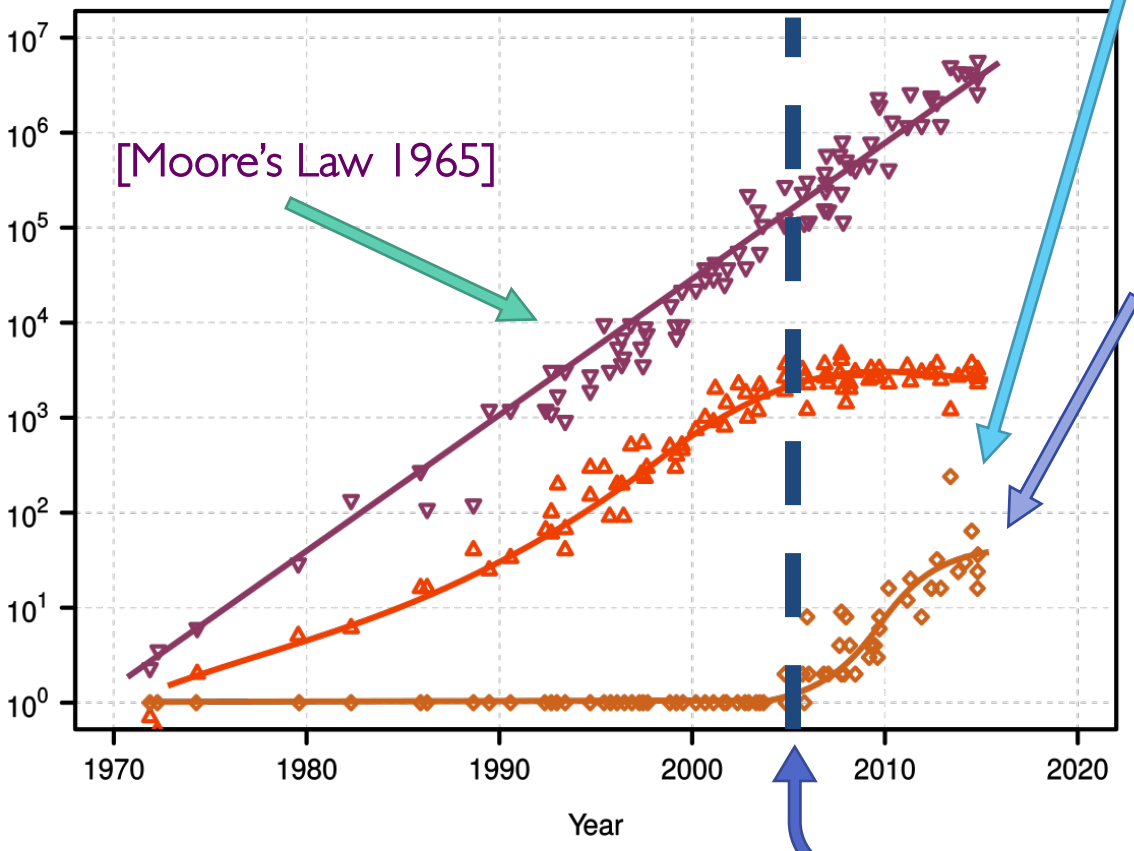


Operating Systems are at the heart of it all …

Technology Trends

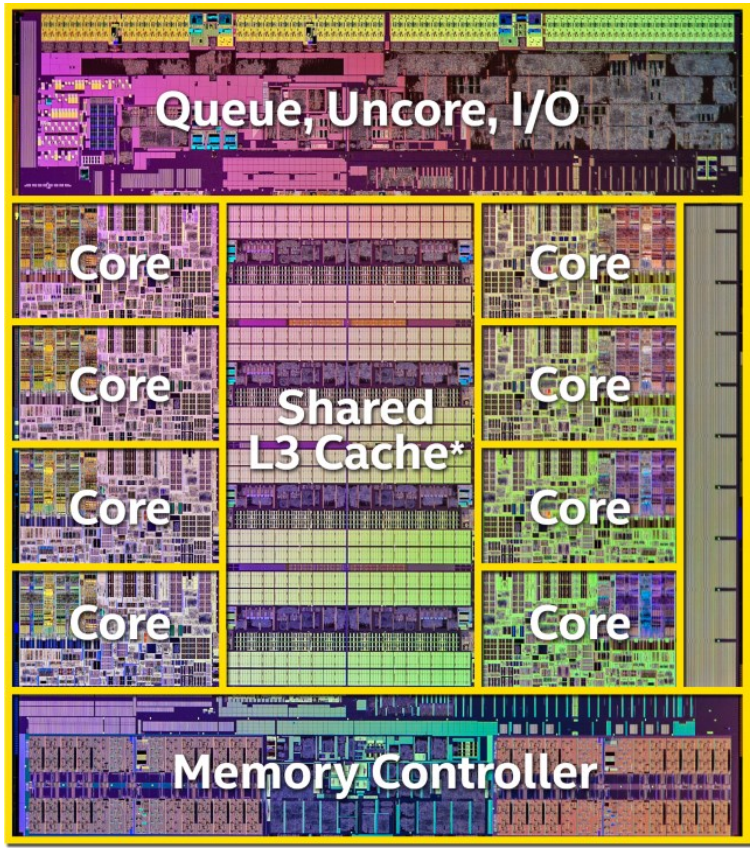
How do we program these?

Parallelism must be exploited at all levels



Modern Processors

Intel Haswell E

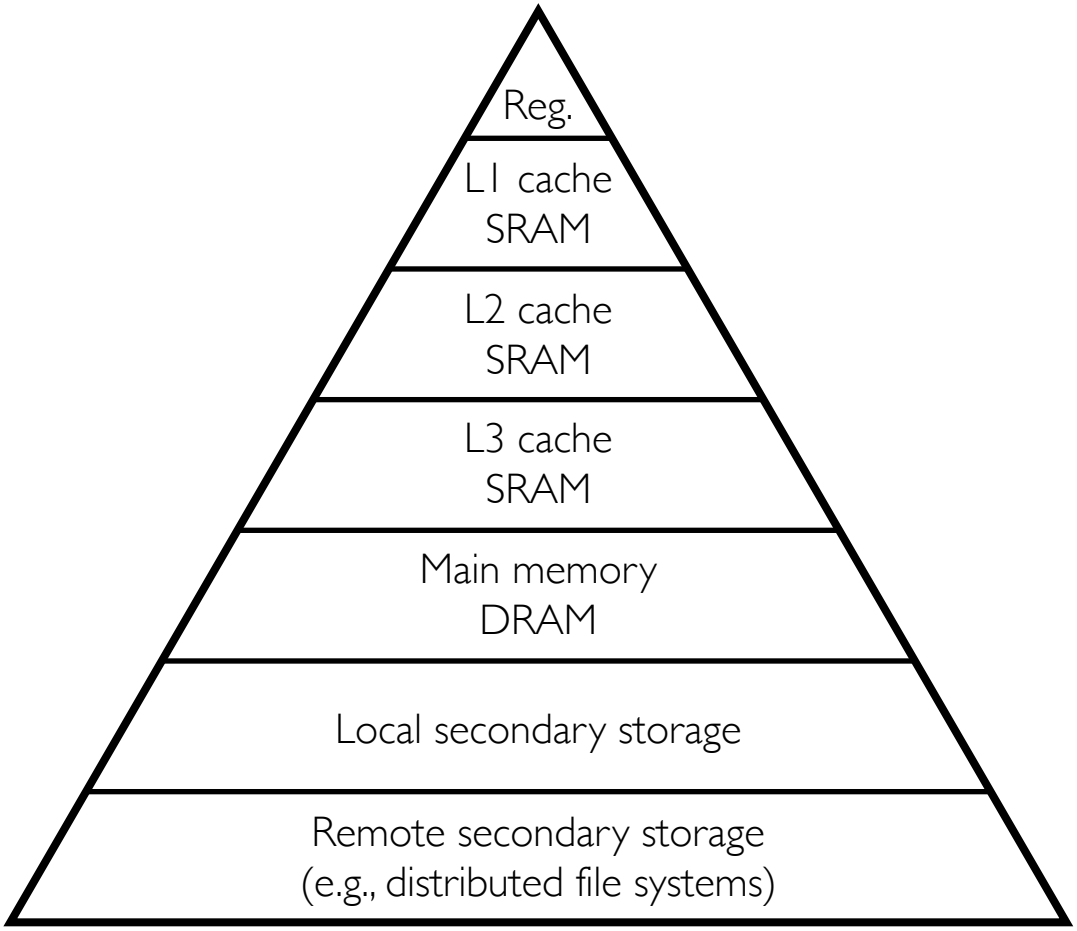


- Intel Xeon Platinum 9282
  - 14nm processor
  - 56 cores, 112 threads
  - 1.75MB data and ins. L1 cache
  - 56MB L2 cache
  - 77MB shared L3 cache
  - 8B transistors
- AMD EPYC 7H12
  - 7nm processor
  - 64 cores, 128 threads
  - 2MB data and ins. L1 cache
  - 32MB L2 cache
  - 256MB shared L3 cache
  - 4.8B transistors

Memory Hierarchy

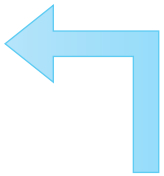
Smaller
Faster
Costlier (per byte)

Larger
Slower
Cheaper (per byte)



Numbers Everyone Should Know [Jeff Dean, 2009]

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns



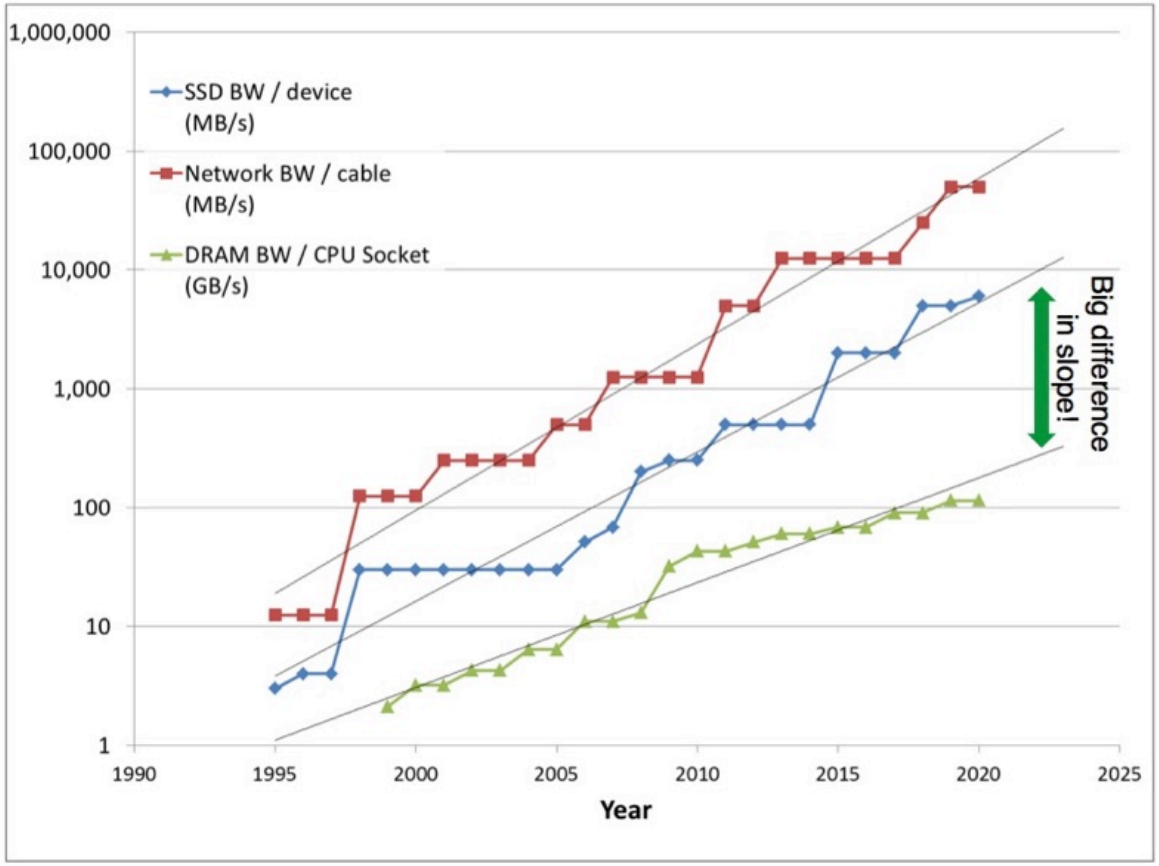
Key stroke ~100 ms

Network, IO, and Memory Bandwidth Trends

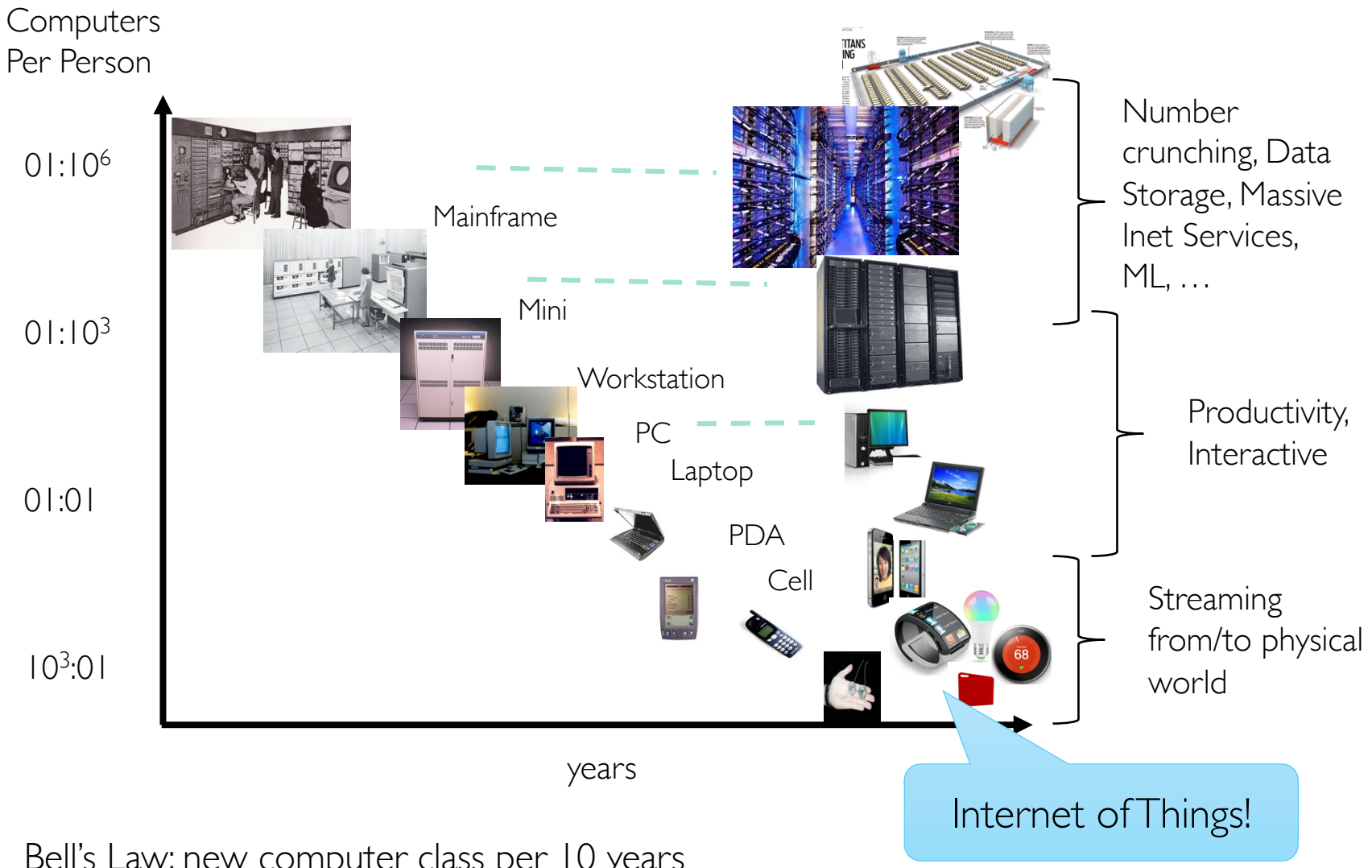
Network, Storage, & DRAM trends

Log scale

- Use DRAM Bandwidth as a proxy for CPU throughput
- Reasonable approximation for DMA and poor cache performance workloads (e.g. Storage)



People to Computer Ratio Trend



Bell's Law: new computer class per 10 years

Early vs Current Operating Systems

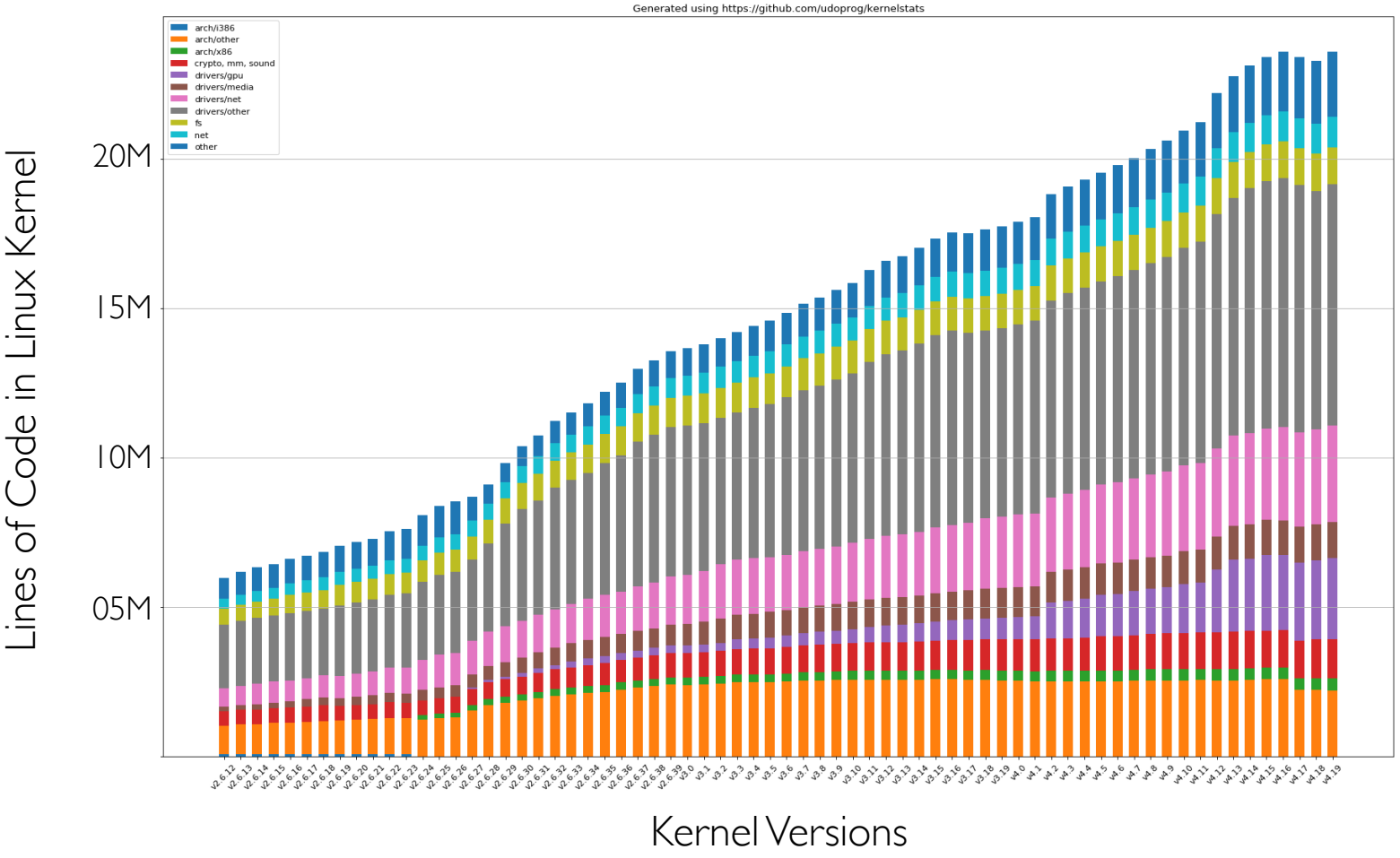
- One user/application at a time
  - Had complete control of hardware
  - OS was runtime library
  - Users would stand in line to use the computer
- Batch systems
  - Keep CPU busy by having a queue of jobs
  - OS would load next job while current one runs
  - Users would submit jobs, and wait, and wait, and wait …
- Multiple users on computer at the same time
  - Multiprogramming: run multiple programs at same time
  - Interactive performance: try to complete everyone's tasks quickly
  - As computers became cheaper, more important to optimize for user time, not computer time

Complexity

- Applications consisting of…
  - … a variety of software modules that …
  - … run on a variety of devices (machines) that
    - … implement different hardware architectures
    - … run competing applications
    - … fail in unexpected ways
    - … can be under a variety of attacks

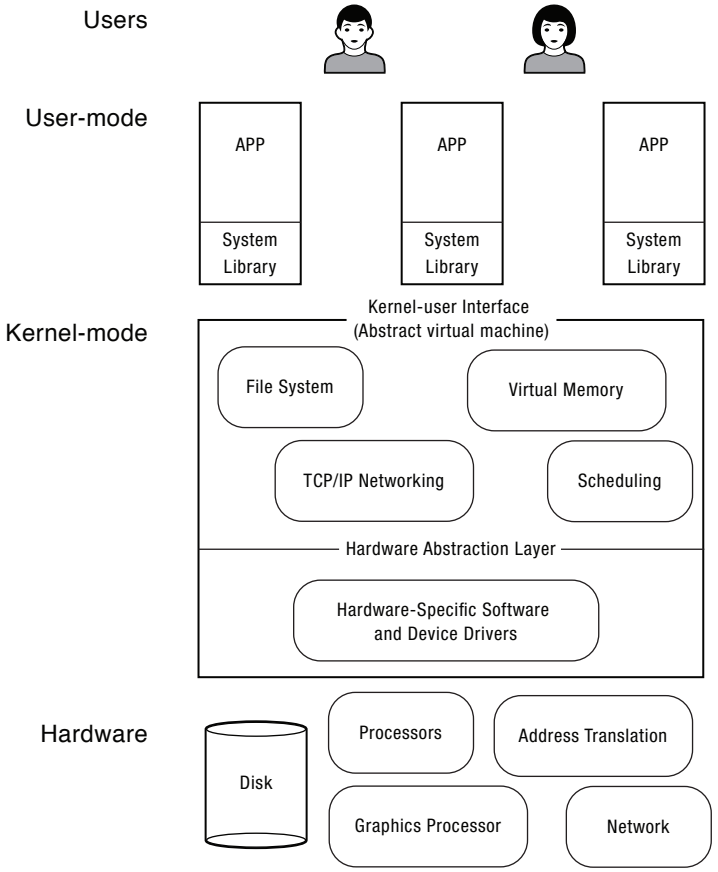
- Not feasible to test software for all possible environments and combinations of components and devices
  - Question is not whether there are bugs but how serious are bugs!

Kernel Complexity



What do OS's do? (revisited)

- Manage hardware resources for users and applications
- Convert what hardware gives into something that application programmers want
- For any OS component, begin by asking two questions:
  - What is hardware interface? (physical reality)
  - What is application interface? (virtual machine)



Virtual Machines

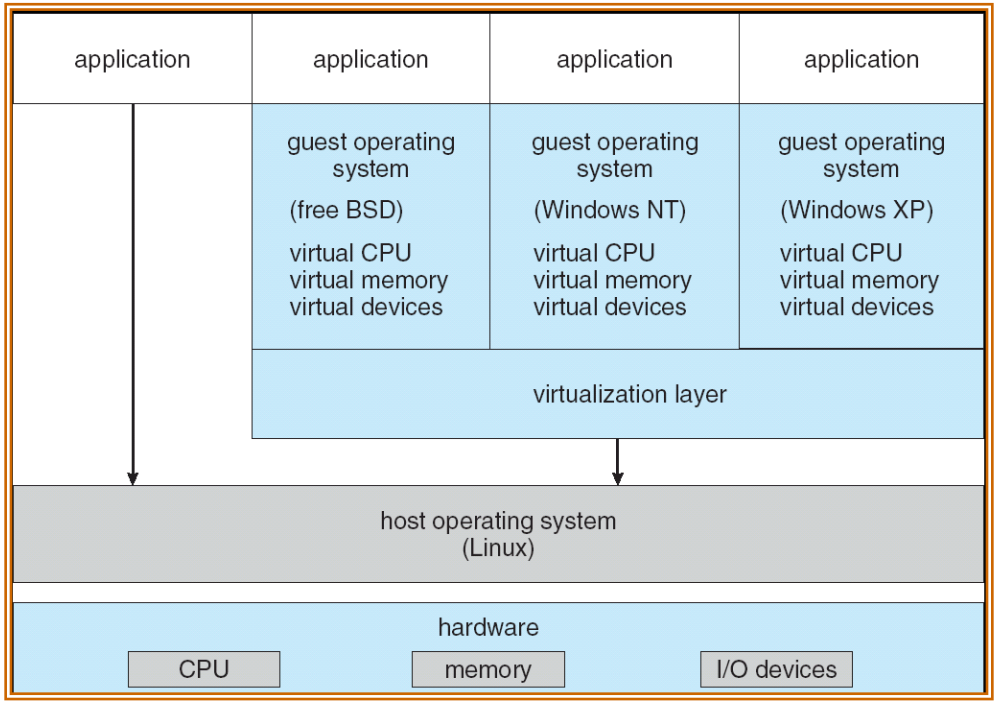
- Software emulation of abstract machine
  - Gives programs illusion that they own entire machine
  - Makes it look like hardware has features programs want
- Two types of "Virtual Machines"
  - Process VM: supports execution of single program; (e.g. OS, Java)
  - System VM: supports execution of entire OS (e.g., VMWare Fusion, Virtual box, Parallels Desktop, Xen)

Process VMs

- Programming simplicity
  - Each process thinks it owns all devices
  - Each process thinks it has all memory/CPU time
  - 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

System Virtual Machines: Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs

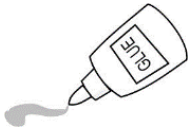


What is an OS? (revisited)

- Referee
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications

- Illusionist
  - Each application appears to have entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

- Glue
  - Libraries, user interface widgets, etc.



What is an OS, … 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?

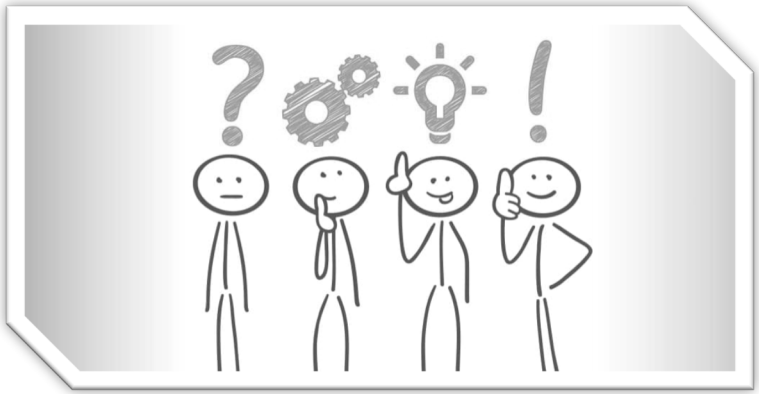
Operating System Definition

- No universally accepted definition
- "Everything vendors ship when you order OS" is good approximation
  - But varies wildly
- "The one program running at all times on computer" is kernel
  - Everything else is either system program (ships with OS) or application program

Summary

- OS's provide virtual machine abstraction to handle diverse hardware
  - OS's simplify application development by providing standard services
- OS's coordinate resources and protect users from each other
  - OS's can provide array of fault containment, fault tolerance, and fault recovery
- SE350 combines ideas and concepts from many other areas of computer science and engineering
  - Languages, data structures, hardware, and algorithms

Questions?



Acknowledgment

- Slides by courtesy of Anderson, Culler, Stoica, Silberschatz, Joseph, and Canny