

Operating Systems (Honor Track)

What is an Operating System?

Xin Jin

Spring 2023

Course Staff

- Instructor: Xin Jin
 - Research interests: Computer systems
 - » Cloud computing
 - » Software-defined datacenters
 - » Machine learning systems



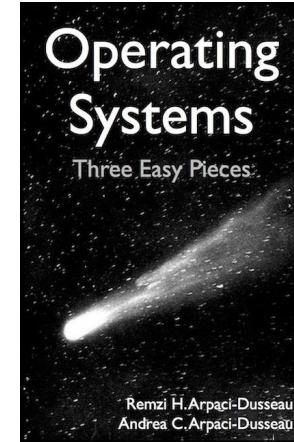
- Teaching assistants



Yinmin Zhong

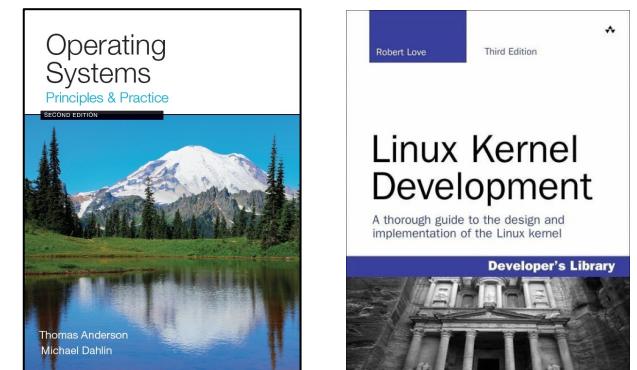
Infrastructure & Recommended Readings

- Infrastructure
 - Website: <https://pku-os.github.io/sp23>
 - Piazza: <https://piazza.com/pku.edu.cn/spring2023/04834490>
 - Course feedback: <https://www.wjx.cn/vm/t5X3TwX.aspx>
 - Office hour: Tuesday 4-5pm, Yanyuan Building 811, or by appointment



- Recommended textbook: Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau

- Available for **free** online (<https://pages.cs.wisc.edu/~remzi/OSTEP>)
 - Try to keep up with materials in book as well as lectures



- Supplementary Materials

- Operating Systems: Principles and Practice, 2nd Edition, by Anderson and Dahlin
 - Linux Kernel Development, 3rd edition, by Robert Love

Preparing Yourself for this Class

- The assignments will require you to be very comfortable with programming and debugging C
 - Pointers (including function pointers, void*)
 - Memory Management (malloc, free, stack vs heap)
 - Debugging with GDB
- You will be working on a larger, more sophisticated code base than anything you've likely seen in other courses!
- Others
 - C tutorial (<https://cs162.org/ladder/>)
 - Markdown tutorial (<https://www.markdowntutorial.com/zh-cn/>)

Goals

- **Prepare** students for advanced study and research in computer systems, by providing the necessary foundation and context.
- **Enable** students to understand the internal core logic of the operating system in depth and have a chance to design and implement their own operating systems.
- **Empower** students to write and debug large programs, design and implement useful abstractions and especially practice their ability to write and debug multi-threading programs.

Class Workload

- **Lectures**
 - Core concepts and principles of operating systems, which form the foundation of computer systems in general (based on Berkeley CS 162)
 - Recent developments and frontiers in computer systems (new in this course)
 - » Big data systems: Google's three papers (GFS, MapReduce, Bigtable), Hadoop, Spark, etc.
 - » Machine learning systems: TensorFlow, PyTorch, Ray, etc.
 - » Cloud computing: resource disaggregation, serverless computing, etc.
 - » We will read and discuss interesting research papers in computer systems
 - Guest lectures from both academia and industry
- **Programming assignments: VERY CHALLENGING**
 - Design and implement your own operating systems
 - Based on Pintos from Stanford (used by OS courses at Stanford, Berkeley, CMU, etc.)
 - » We are working on a new instructional OS based on Rust and RISC-V. Join us if interested 😊
- **Two exams**
 - Midterm, final

Syllabus

- OS Concepts: How to Navigate as a Systems Programmer!
 - Process, I/O, networks
- Concurrency
 - Threads, scheduling, locks, deadlock, scalability, fairness
- Address Space
 - Virtual memory, address translation, protection, sharing
- File Systems
 - I/O devices, file objects, storage, naming, caching, performance, paging, transactions, databases
- Distributed Systems
 - Protocols, RPC, NFS, DHTs, Consistency, Scalability, multicast
- Reliability & Security
 - Fault tolerance, protection, security
- Cloud Infrastructure

Grading

- Participation: 5%
- Programming assignments (5 labs): 40%
 - 5 labs: 4% + 9% + 9% + 9% + 9%
 - Design (design doc) + Implementation (code)
 - Difference from Berkeley CS 162
 - » Done individually
 - » We reorganize the labs
 - » We provide TA sessions that give you enough tips to help you go through the assignments
 - » We do not have other programming assignments in addition to the 5 labs (there are 7 extra programming assignments in Berkeley CS 162)
- Midterm exam: 20%
- Final exam: 35%

Collaboration Policy



Explaining a concept to other students
Discussing algorithms/testing strategies with other students
Searching online for generic algorithms (e.g., hash table)



Sharing code or test cases with other students
Copying OR reading another student's code or test cases
Copying OR reading online code or test cases
Uploading your solutions online (during AND after the course)

We compare all homework submissions against each other and online solutions and will take actions against offenders

Feedback

- Feedback survey: <https://www.wjx.cn/vm/t5X3TwX.aspx>
 - Feel free to send us your feedback about the course.
 - It is available throughout the semester.
 - It is anonymous. A private channel to talk to me.
 - Your suggestions, comments, concerns and questions are **very valuable**.
 - **We will summarize the feedback and make changes to the course.**
 - » Let's make the course better together.

Lecture Goal

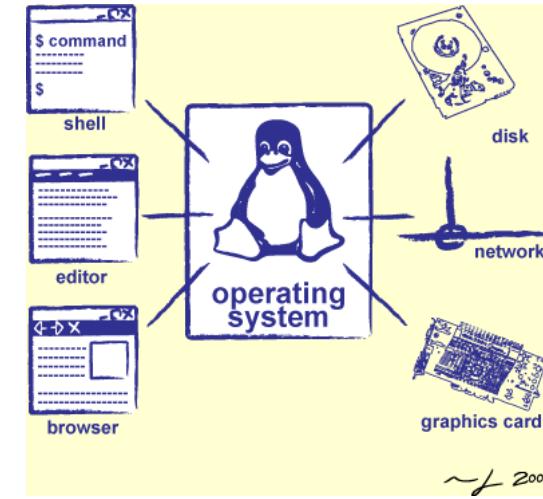
Interactive!!!

Goals for Today

- What is an Operating System?
 - And – what is it not?
- What makes Operating Systems so exciting?

Interactive is important!

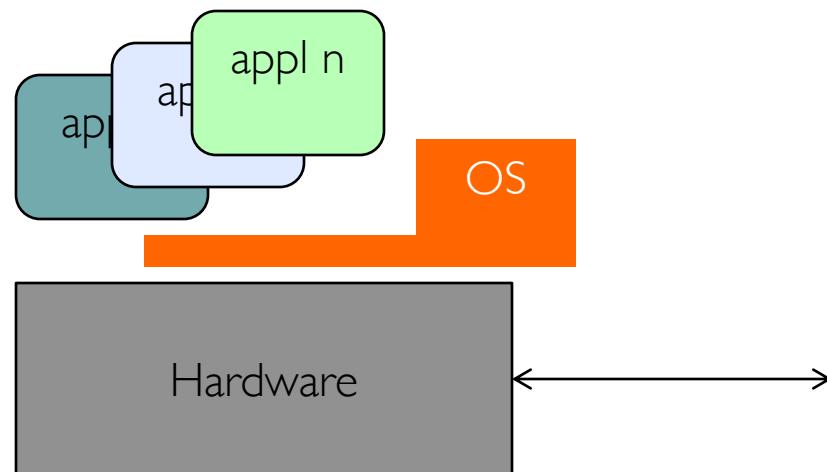
Ask Questions!



Slides courtesy of David Culler, Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

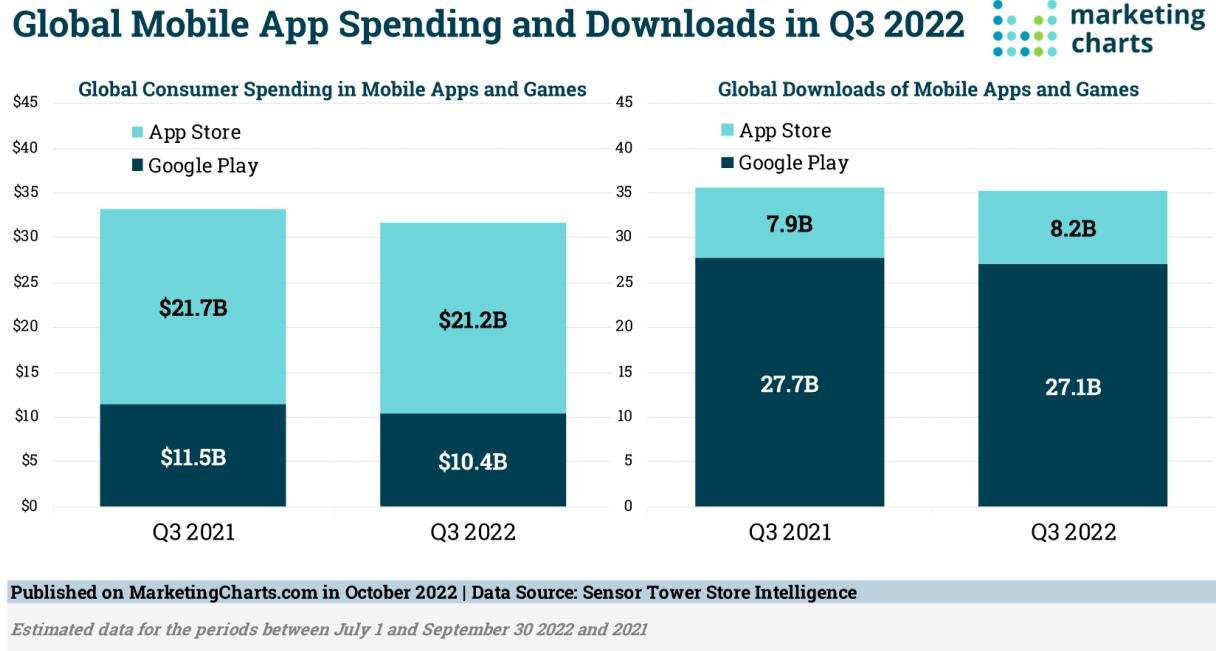
What is an operating system?

- Special layer of software that provides application software access to hardware resources
 - Convenient abstraction of complex hardware devices
 - Protected access to shared resources
 - Security and authentication
 - Communication amongst logical entities

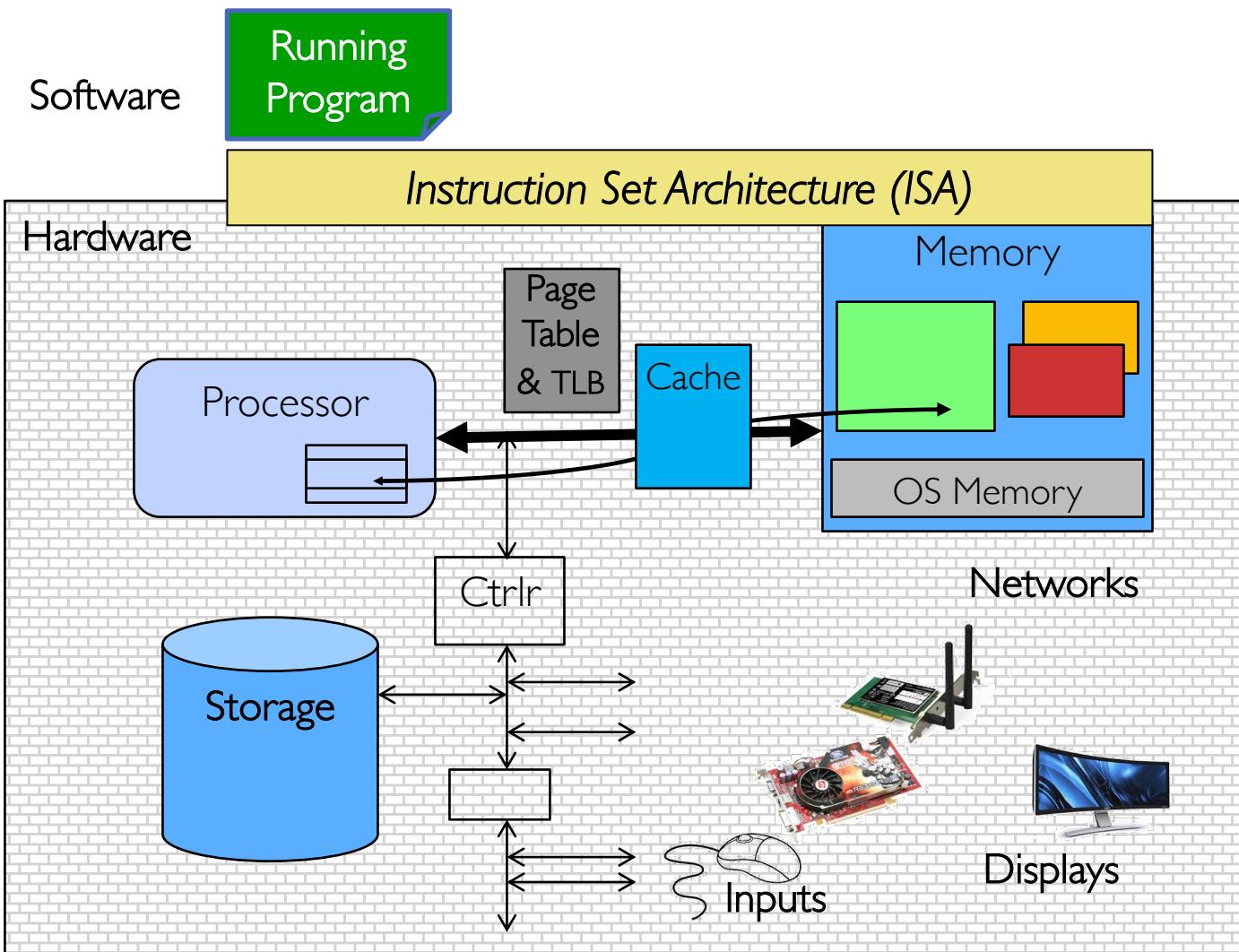


What Does an OS do?

- Provide abstractions to apps
 - File systems
 - Processes, threads
 - VM, containers
 - Naming system
 - ...
- Manage resources:
 - Memory, CPU, storage, ...
- Achieves the above by implementing specific algorithms and techniques:
 - Scheduling
 - Concurrency
 - Transactions
 - Security
 - ...



Hardware/Software Interface



What you learned in
Introduction to
Computer Systems (ICS)

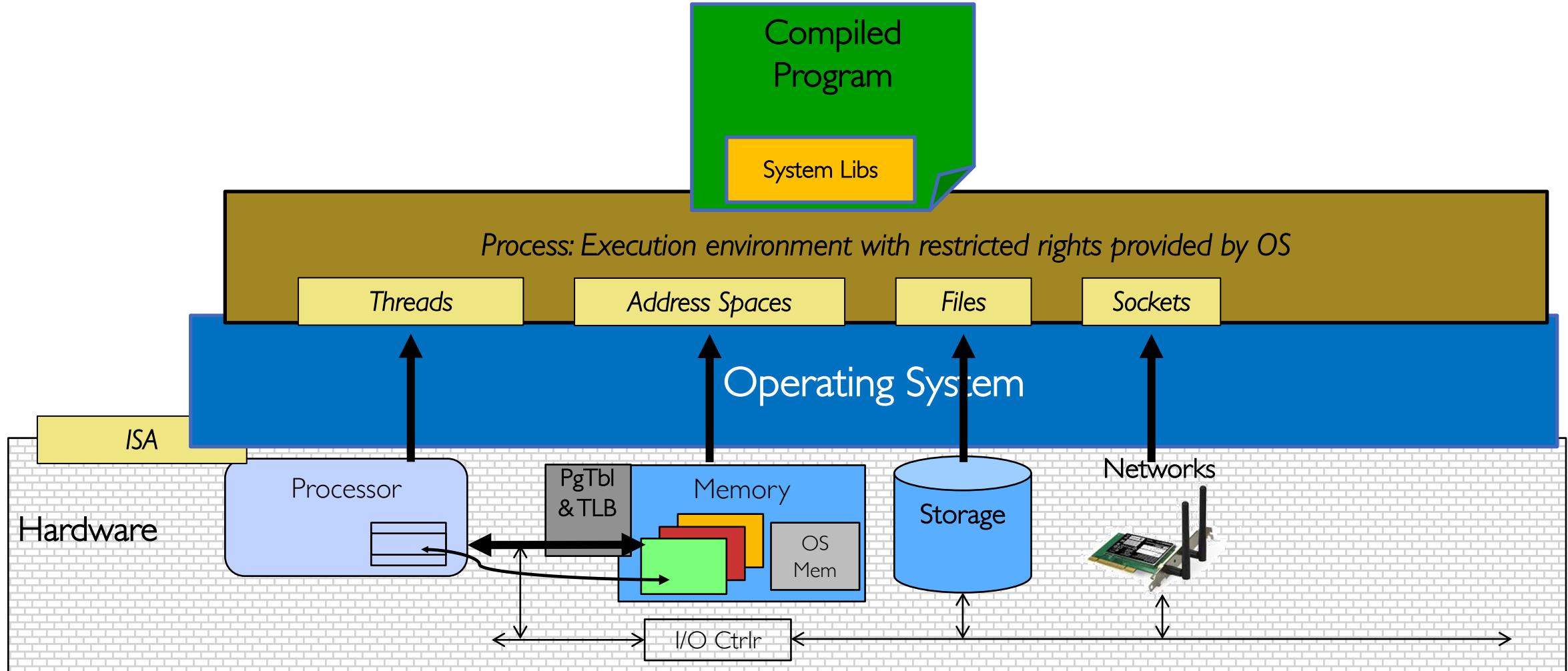
The OS *abstracts* these
hardware details from the
application

What is an Operating System?

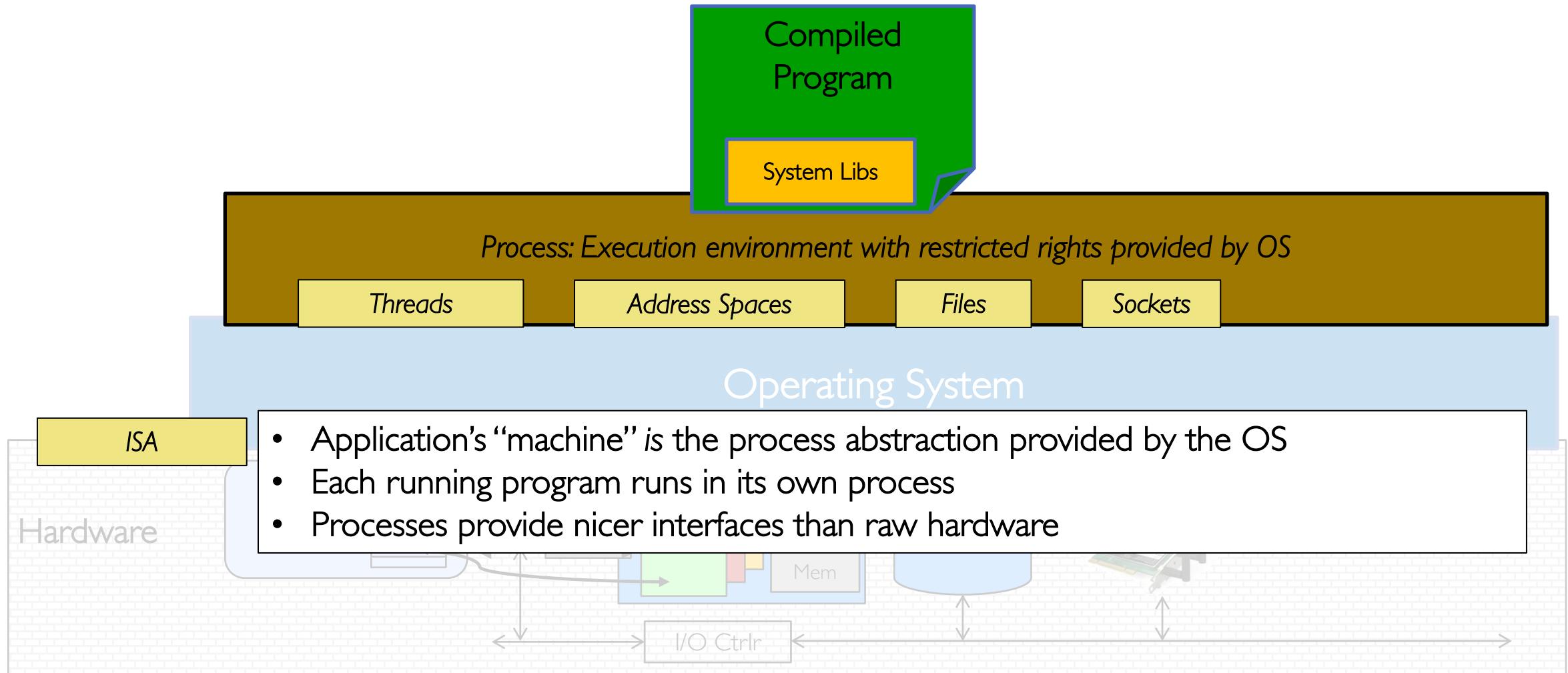


- Illusionist
 - Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

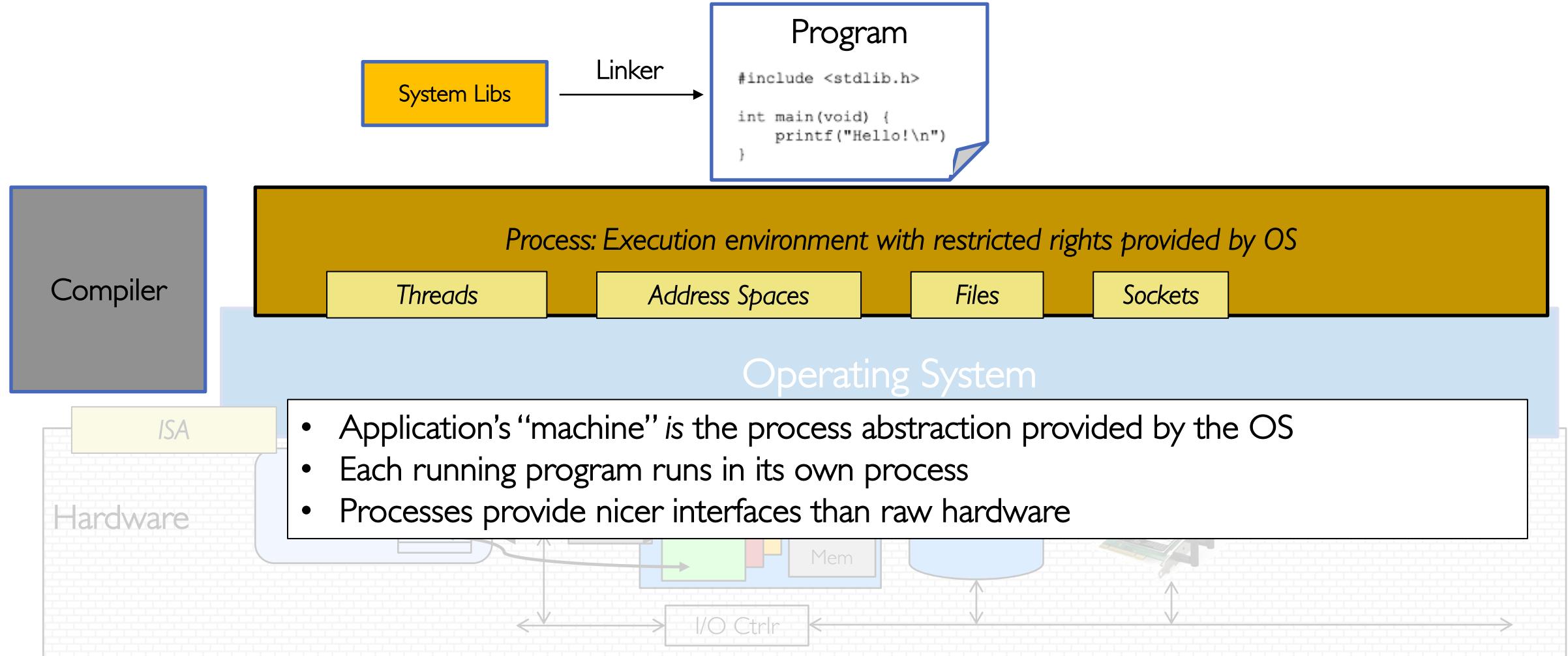
OS Basics: Virtualizing the Machine



Compiled Program's View of the World



System Programmer's View of the World

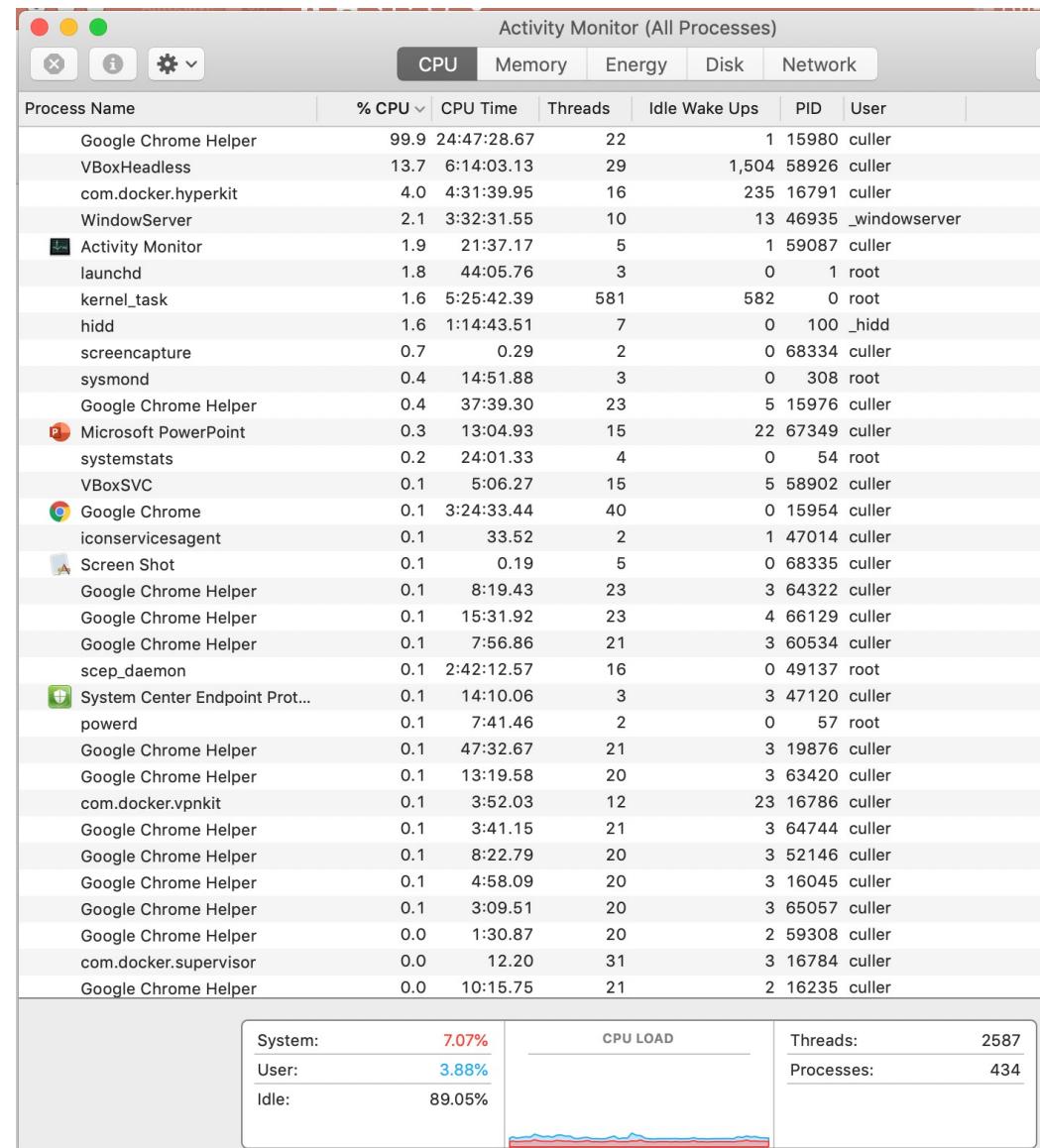


What's in a Process?

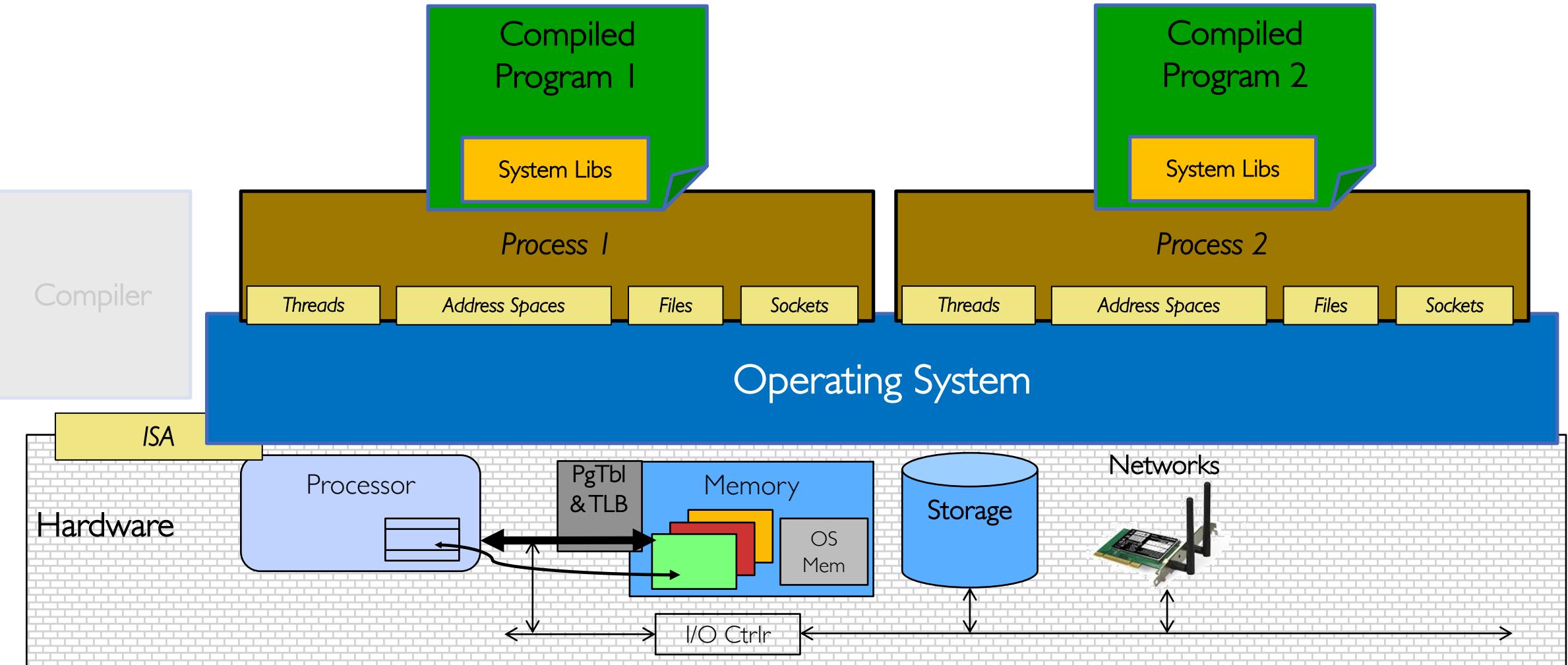
A process consists of:

- Address Space
- One or more threads of control executing in that address space
- Additional system state associated with it
 - Open files
 - Open sockets (network connections)
 - ...

For Example...

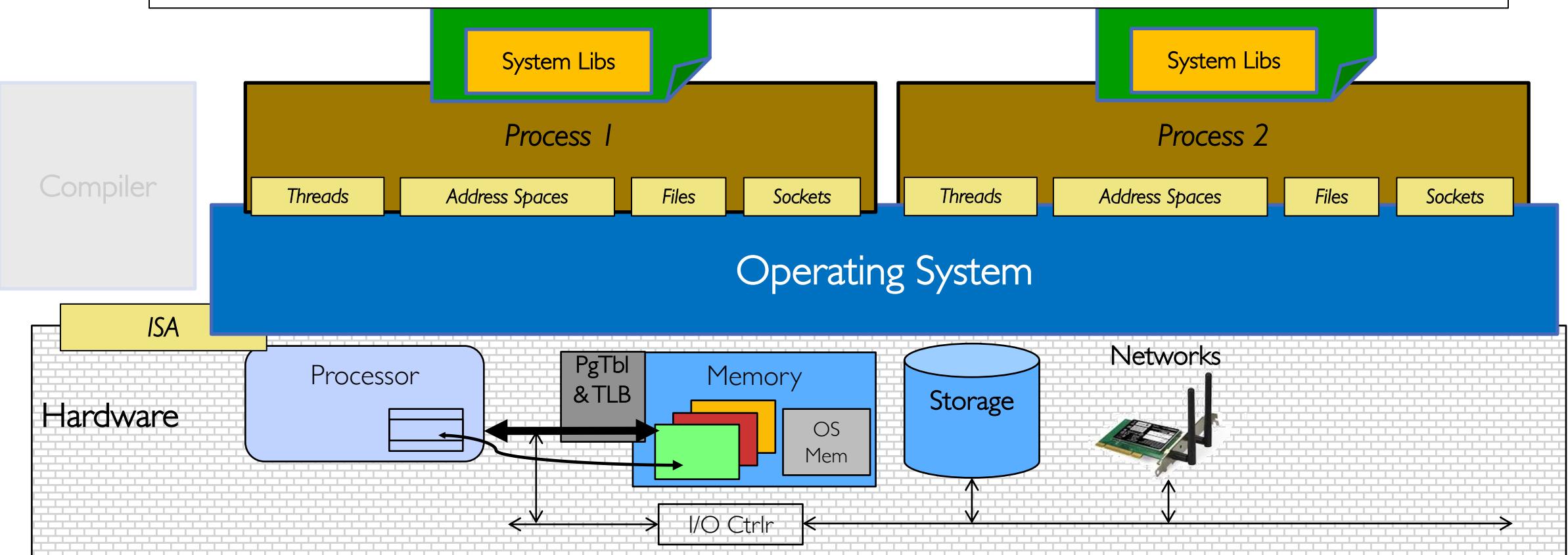


Operating System's View of the World



Operating System's View of the World

- OS translates from hardware interface to application interface
- OS provides each running program with its own process



What is an Operating System?



- Referee

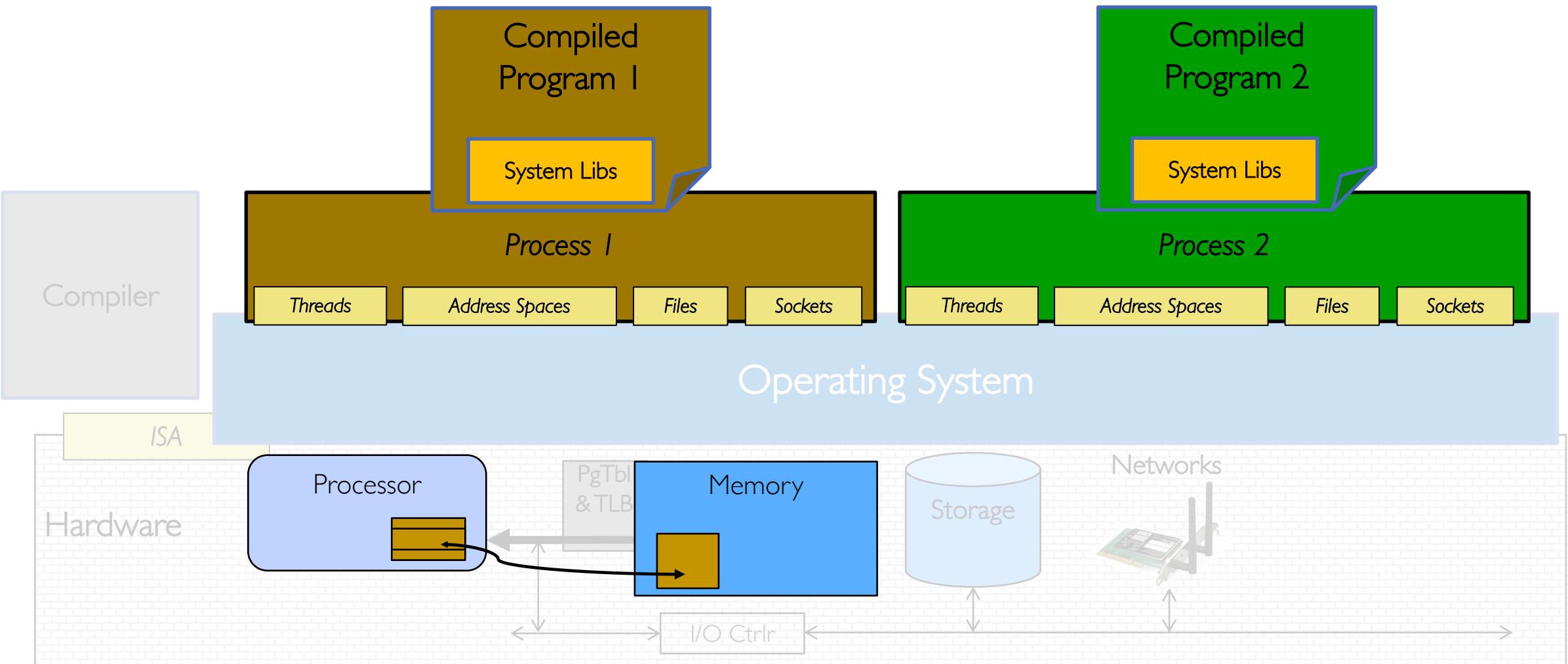
- Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication



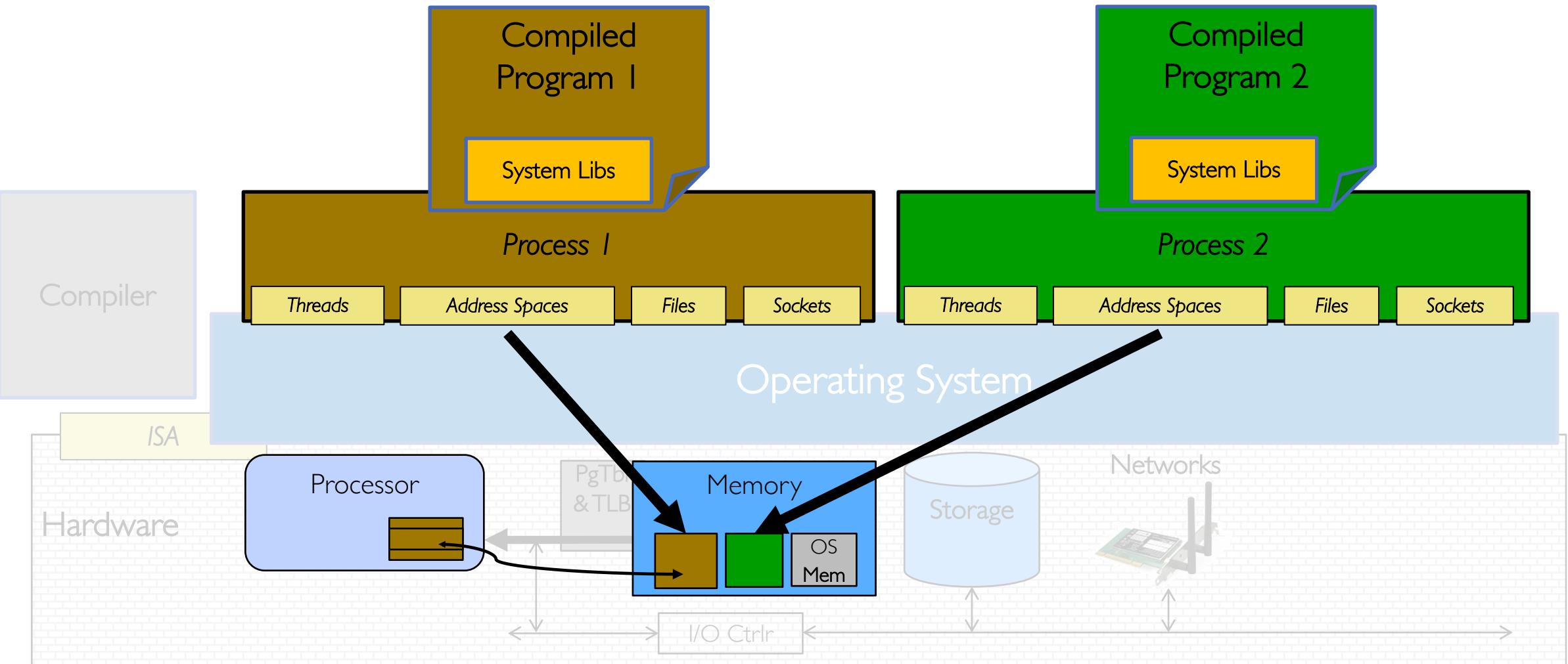
- Illusionist

- Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

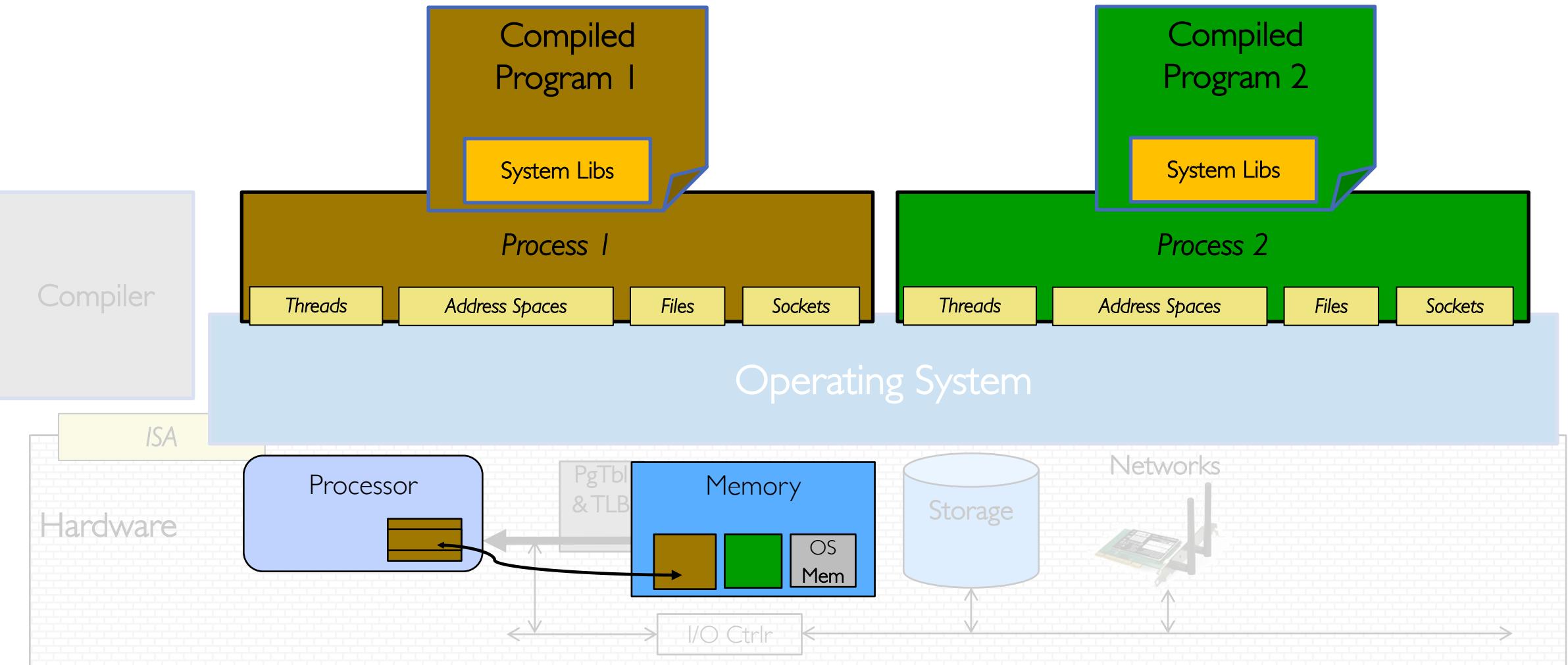
OS Basics: Running a Process



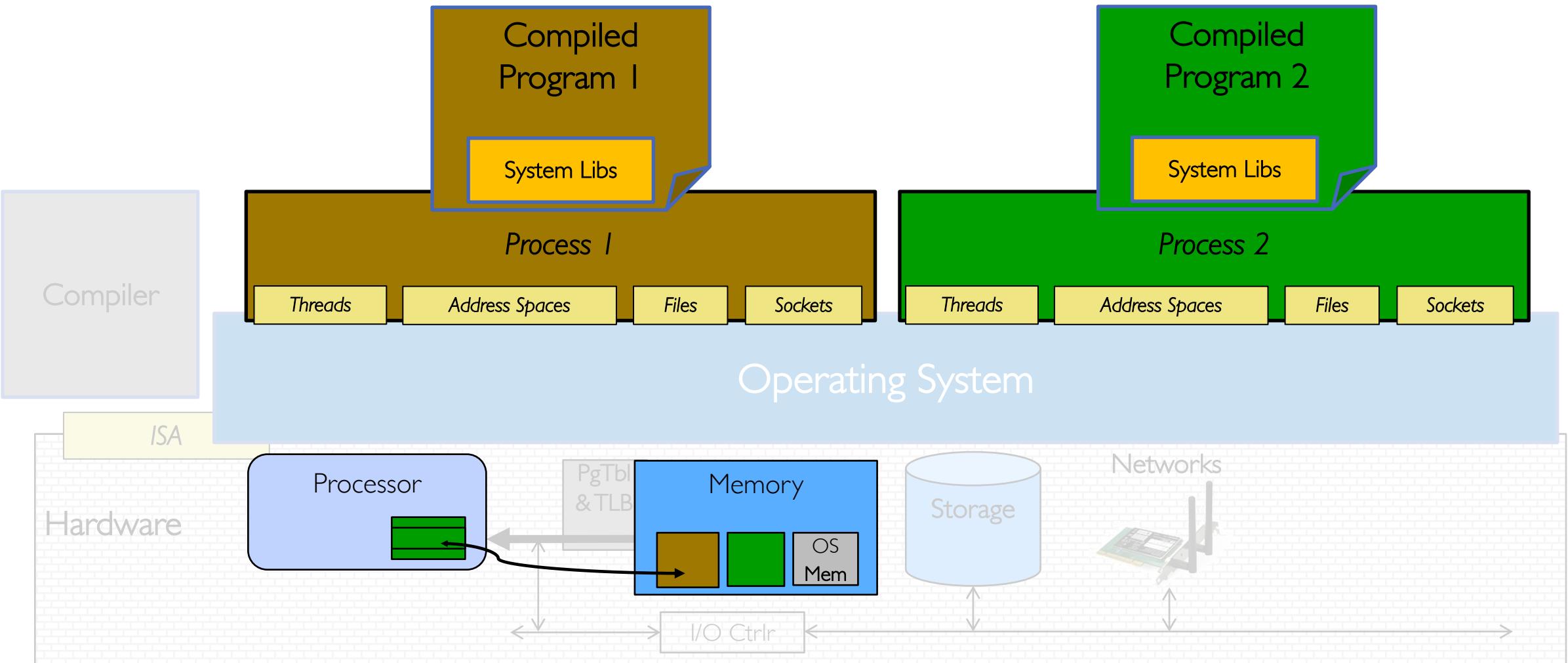
OS Basics: Switching Processes



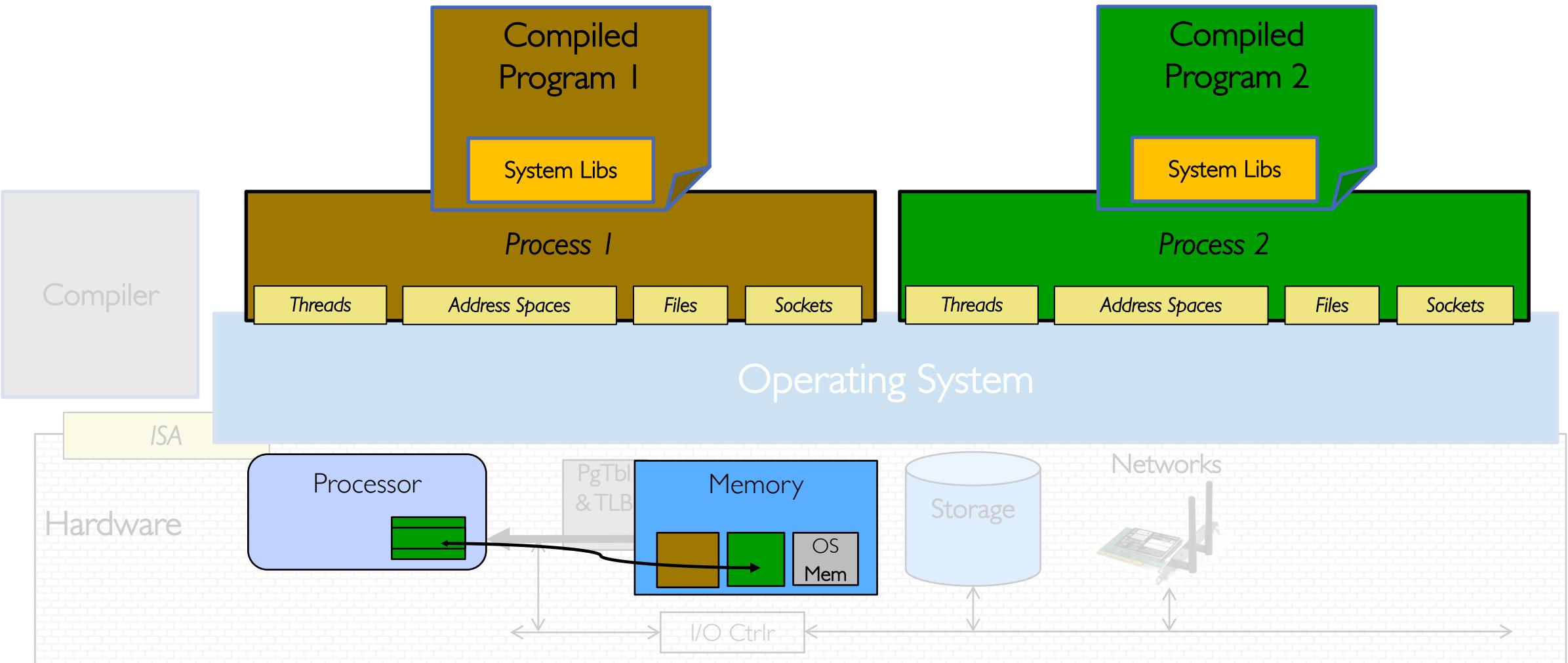
OS Basics: Switching Processes



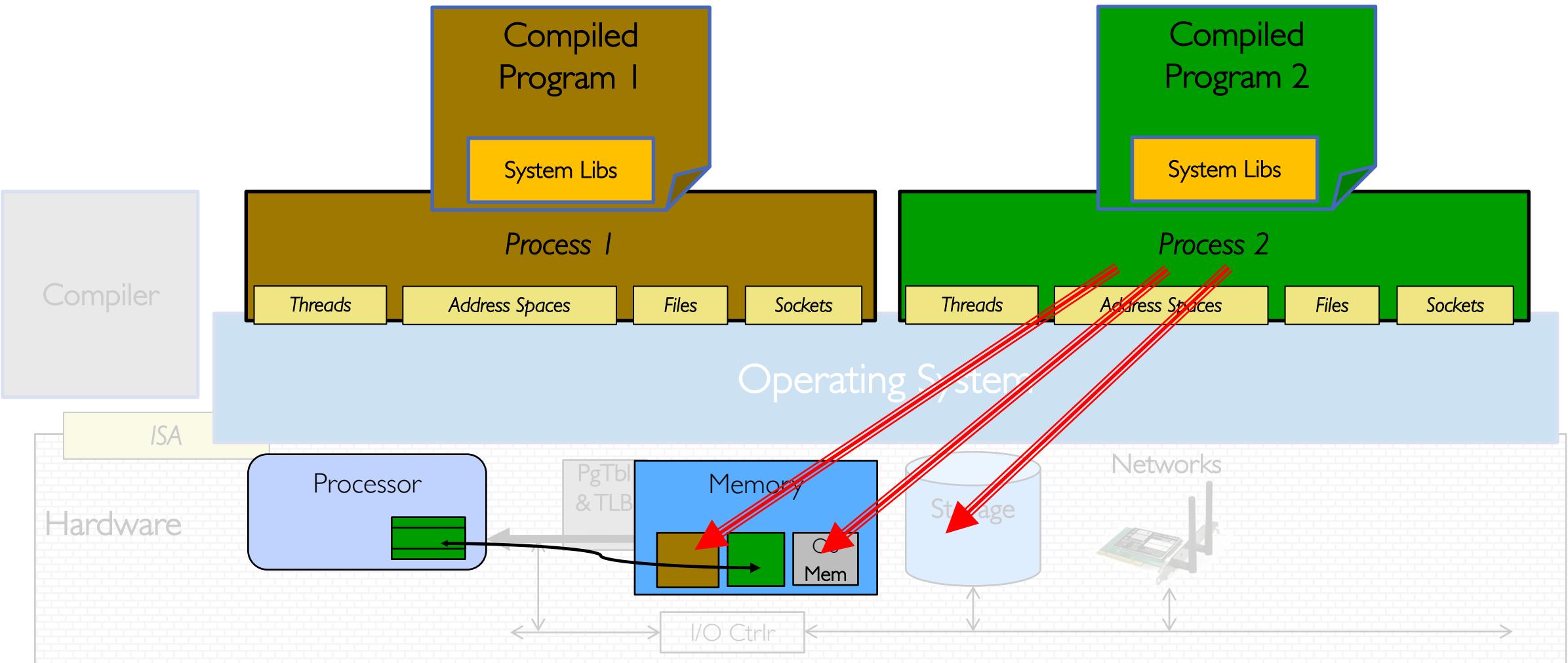
OS Basics: Switching Processes



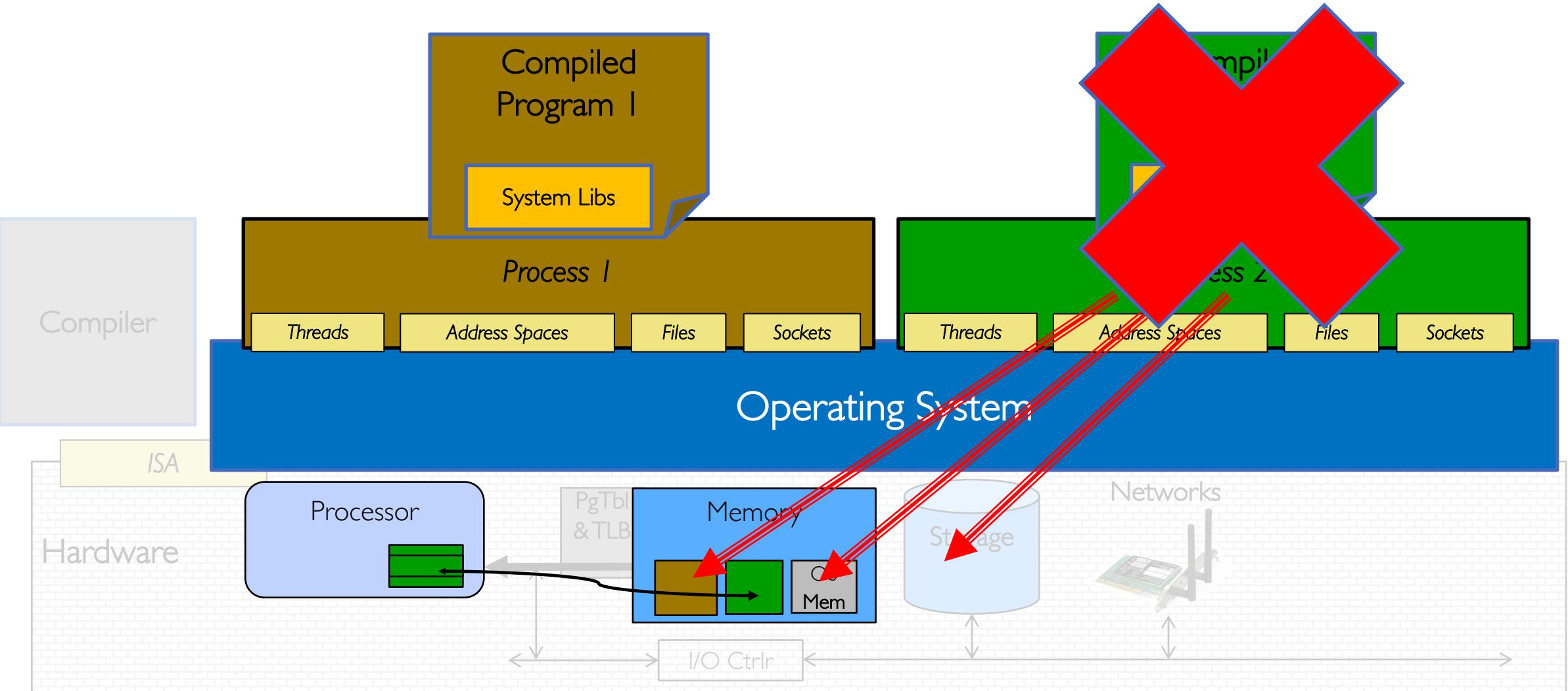
OS Basics: Switching Processes



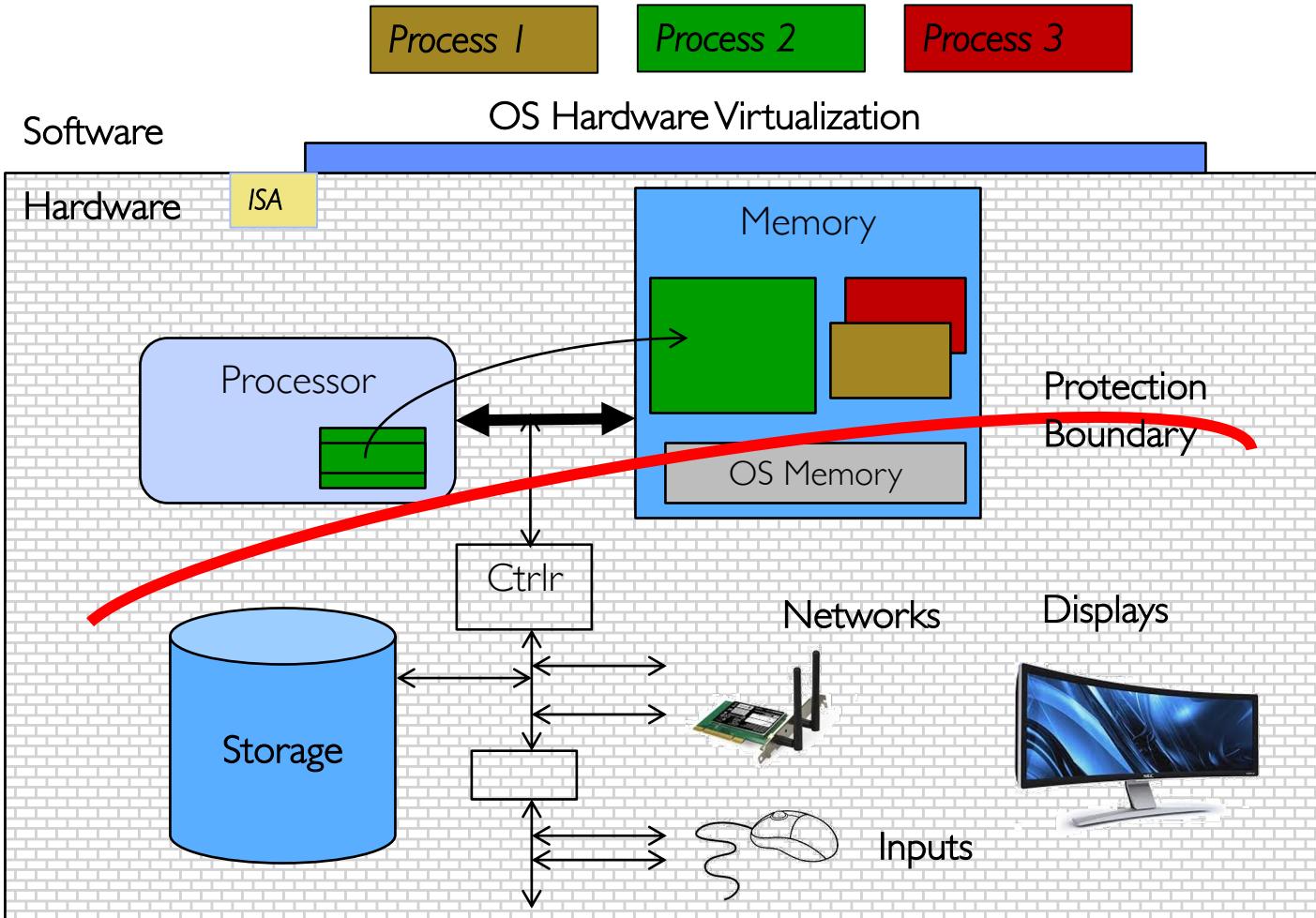
OS Basics: Protection



OS Basics: Protection



OS Basics: Protection



- OS *isolates* processes from each other
- OS *isolates* itself from other processes
- ... even though they are actually running on the same hardware!

What is an Operating System?



- Referee

- Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication



- Illusionist

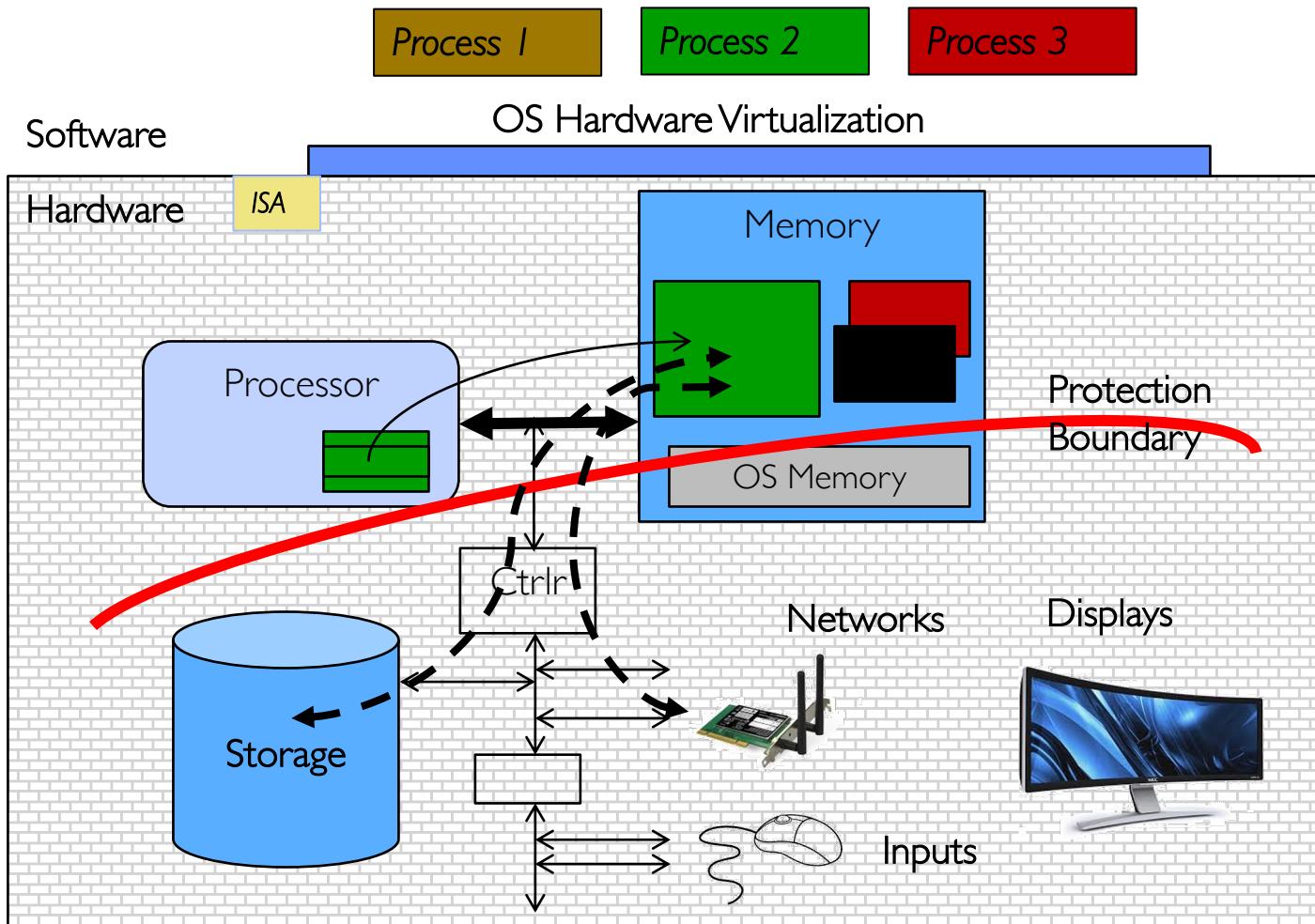
- Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

- Glue



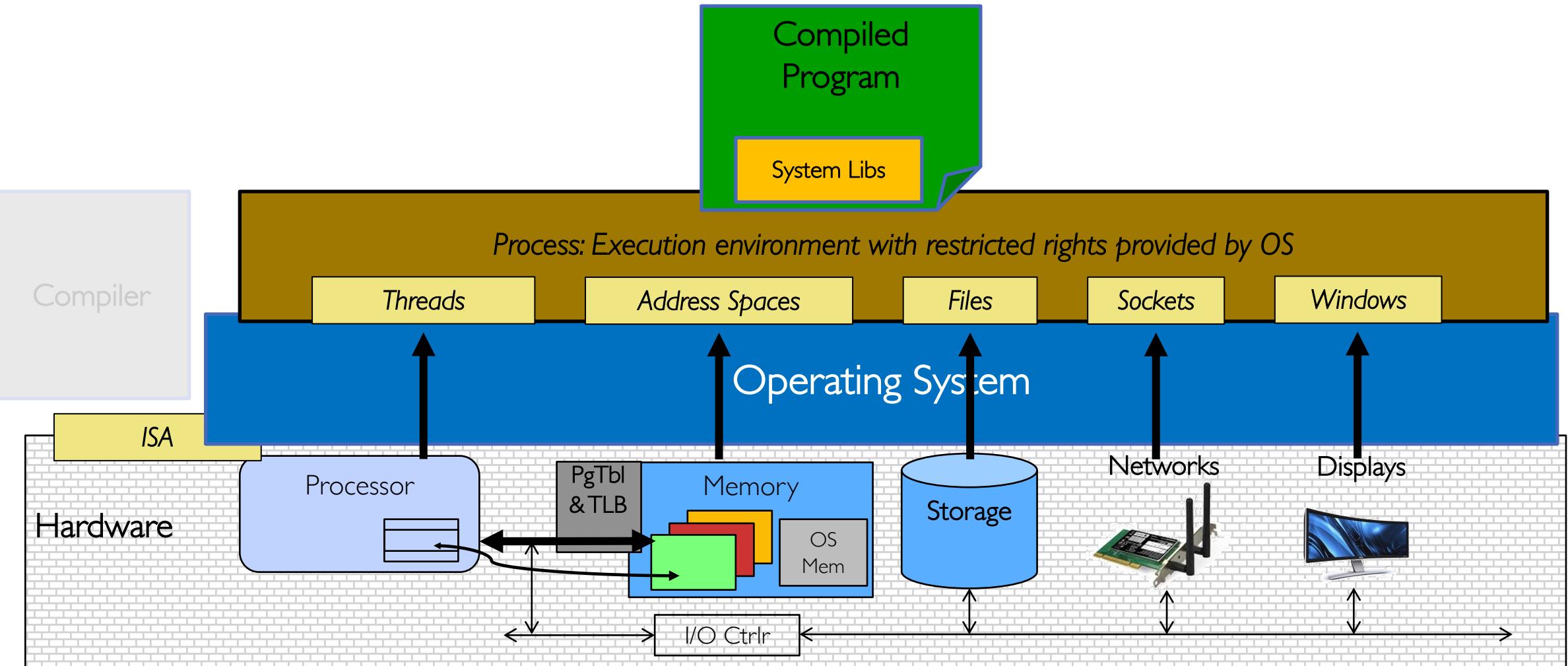
- Common services
 - » Storage, Window system, Networking
 - » Sharing, Authorization
 - » Look and feel

OS Basics: I/O

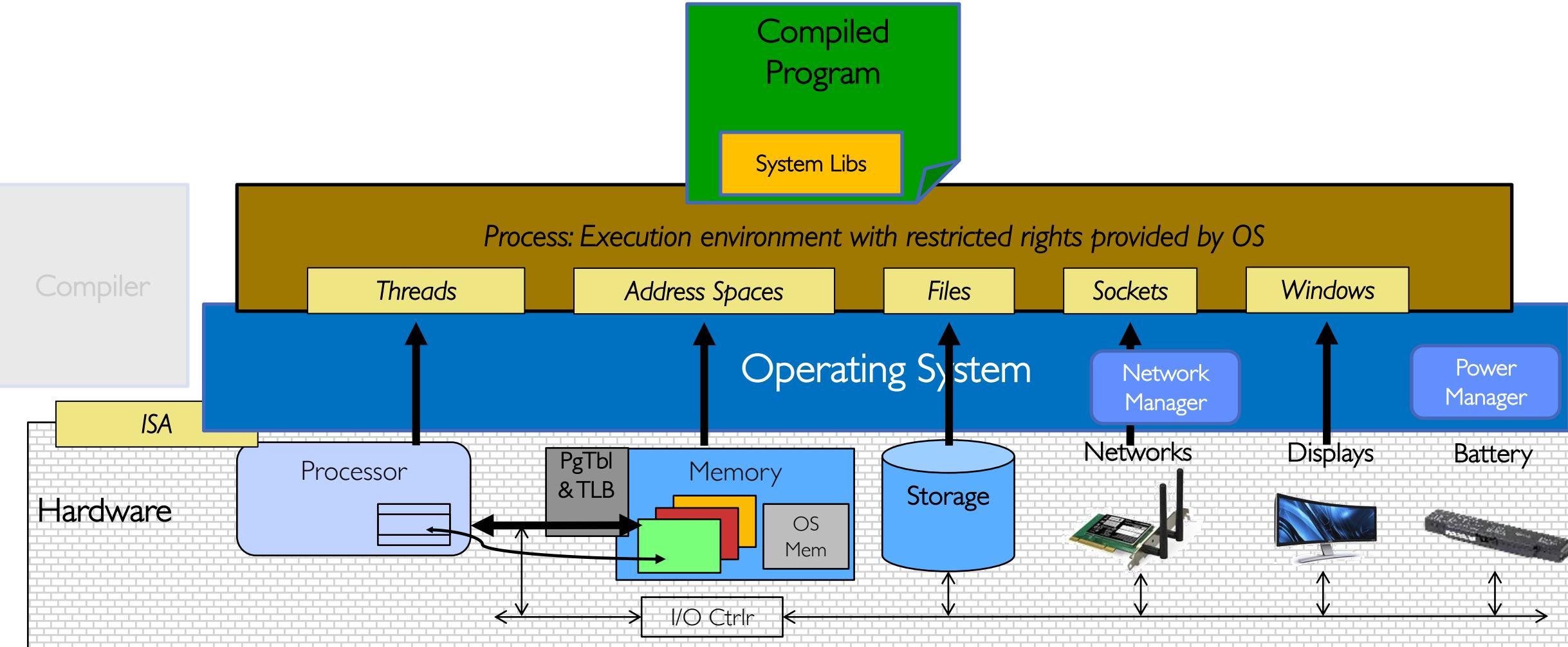


- OS provides common services in the form of I/O

OS Basics: Look and Feel



OS Basics: Background Management



What is an Operating System?



- Referee

- Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication



- Illusionist

- Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

- Glue



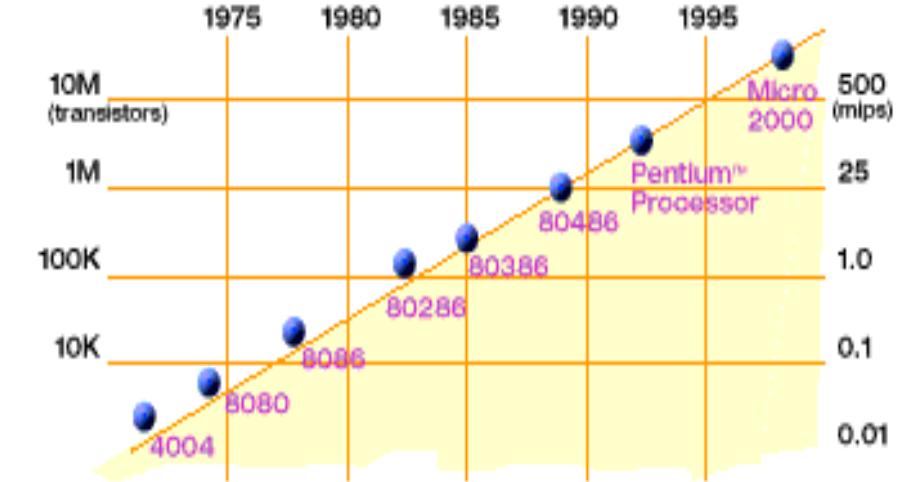
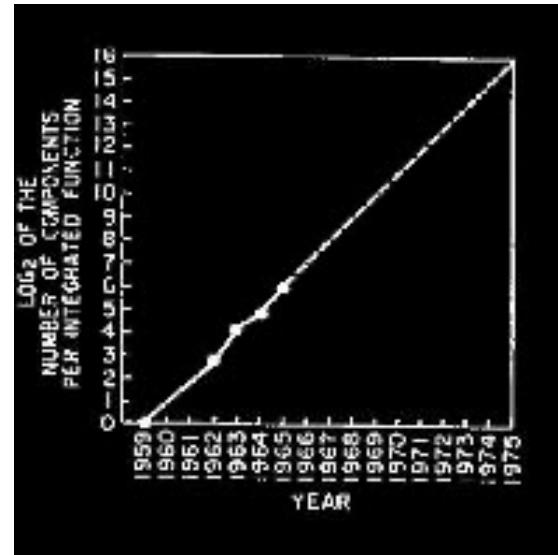
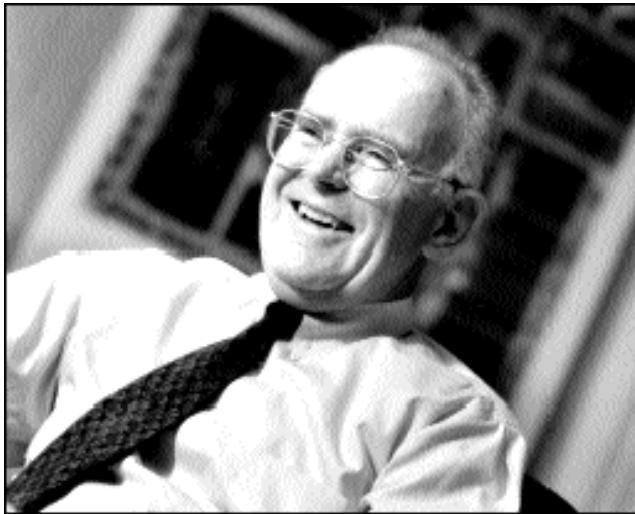
- Common services
 - » Storage, Window system, Networking
 - » Sharing, Authorization
 - » Look and feel

Why take Operating Systems?

- Some of you will actually design and build operating systems or components of them.
 - Perhaps more now than ever
- Many of you will create systems that utilize the core concepts in operating systems.
 - Whether you build software or hardware
 - The concepts and design patterns appear at many levels
- All of you will build applications, etc. that utilize operating systems
 - The better you understand their design and implementation, the better use you'll make of them.

What makes Operating Systems
Exciting and Challenging?

Technology Trends: Moore's Law



2X transistors/Chip Every 2 years
Called "Moore's Law"

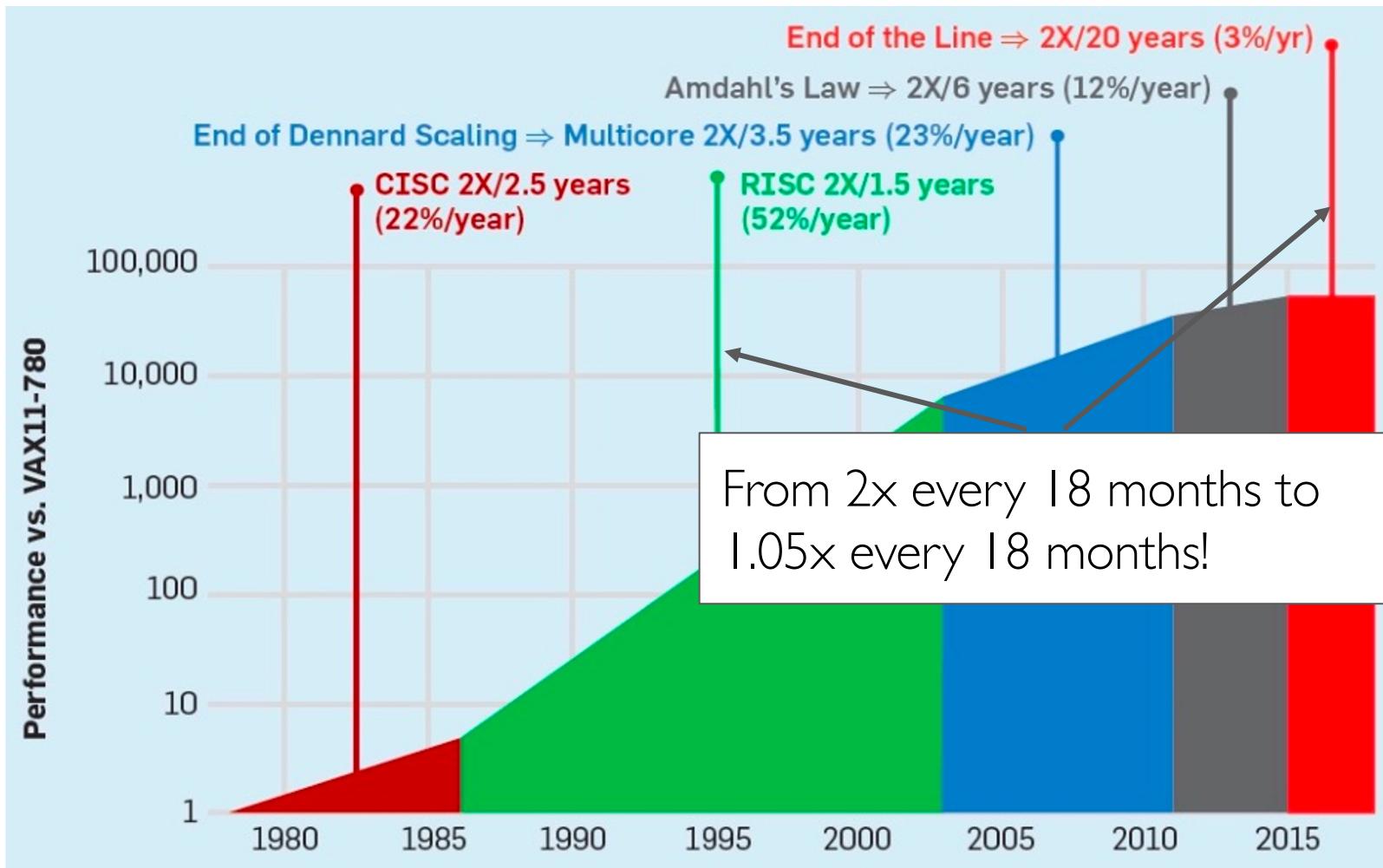
Gordon Moore (co-founder of Intel) predicted in 1965 that the **transistor density** of semiconductor chips **would double roughly every 2 years**

- Microprocessors have become smaller, denser, and more powerful

Corollary: Performance double roughly every 1.5 years (18 months)

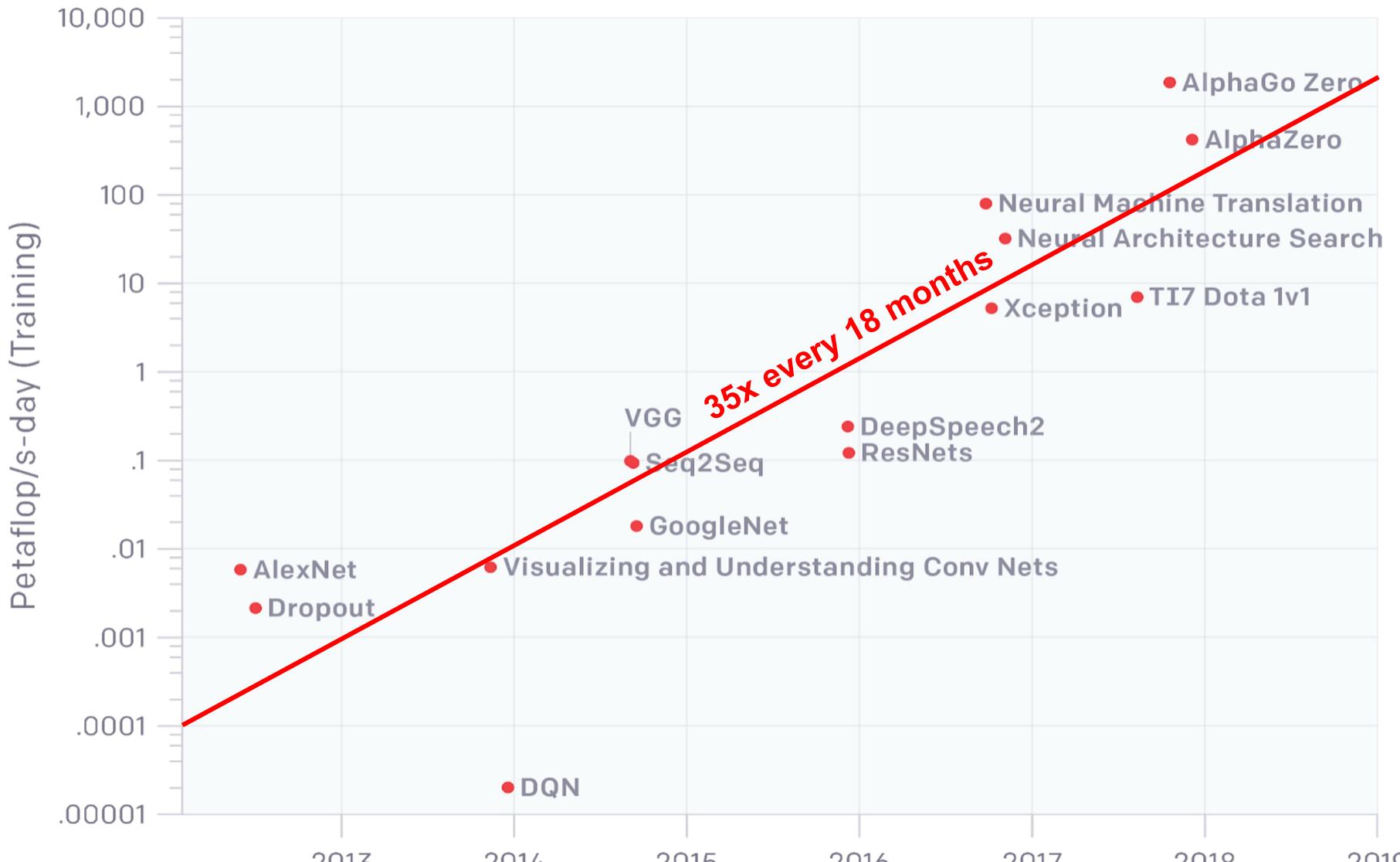
- Faster growth because transistors are closer to each other so electrical signals travel faster

The end of the Moore's Law



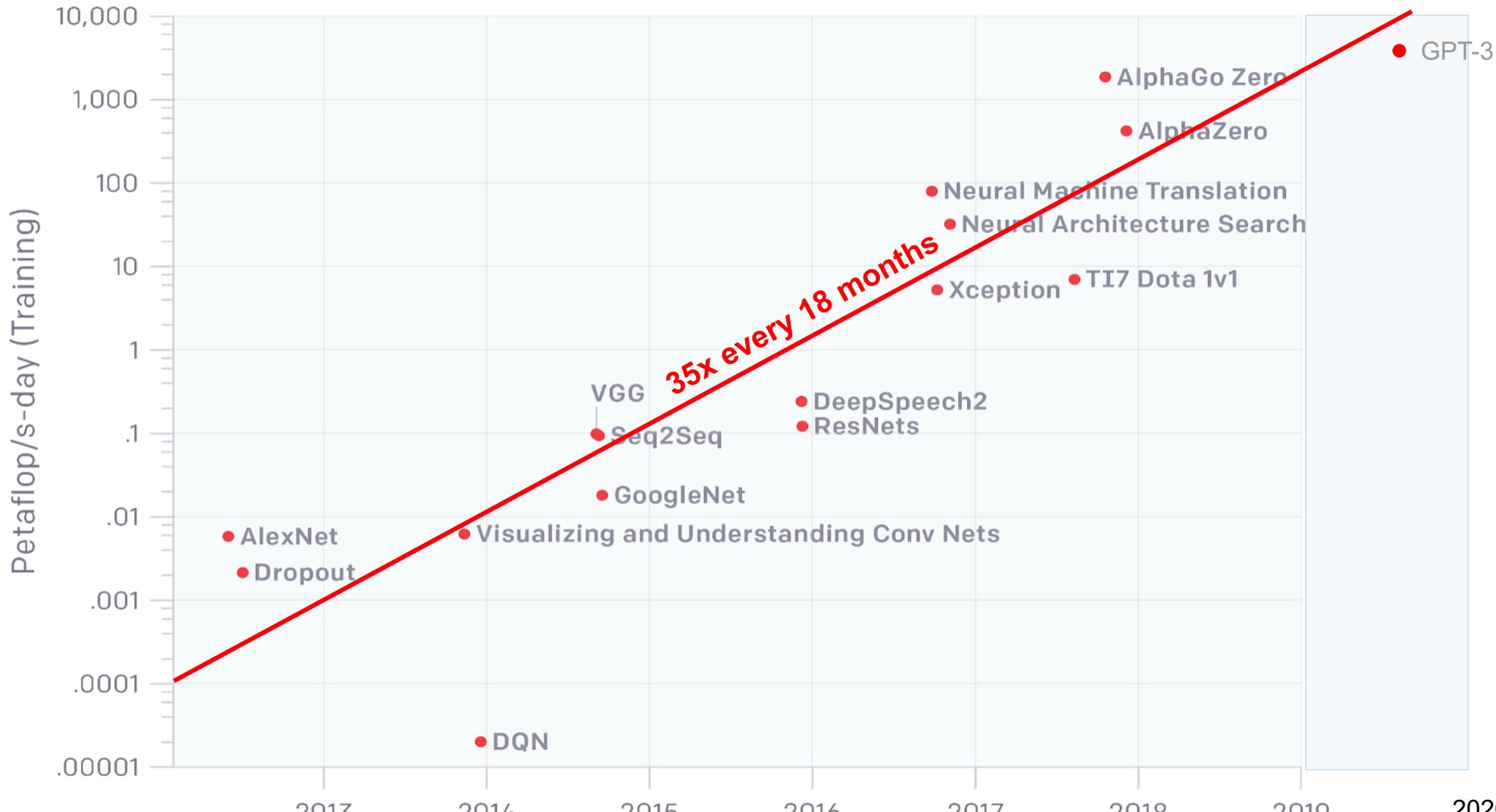
AI Compute demands: 2012 - 2019

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



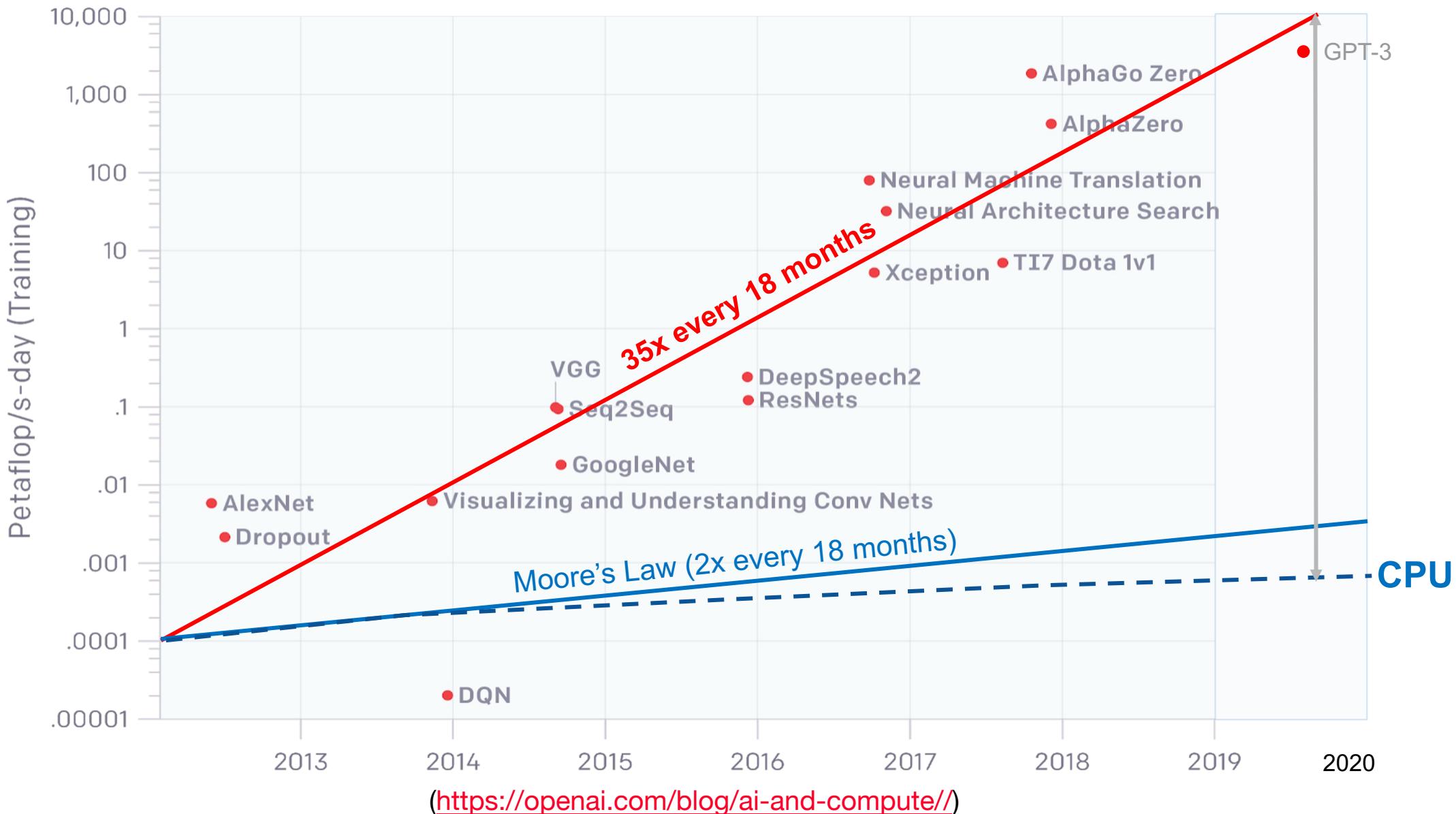
AI Compute demands: 2012 – 2020

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

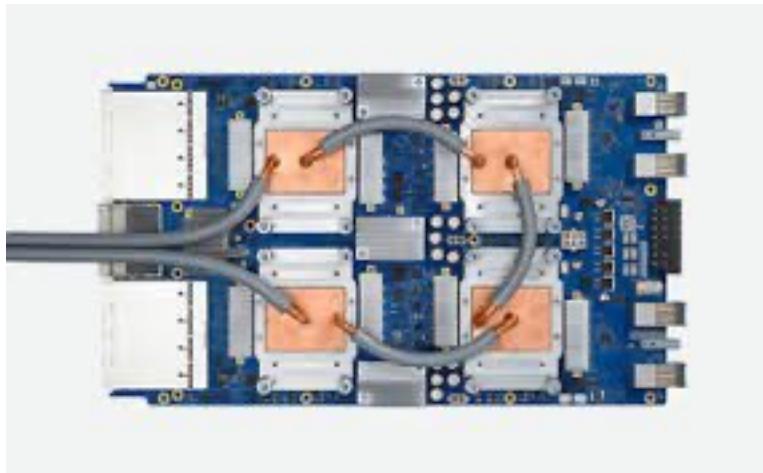


Growing gap between demand and supply

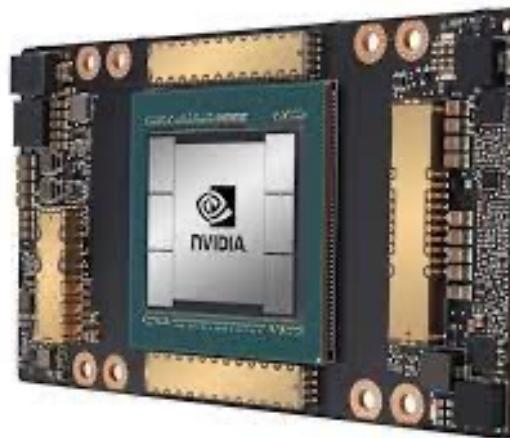
AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



Specialized processors to the rescue?



TPU v3



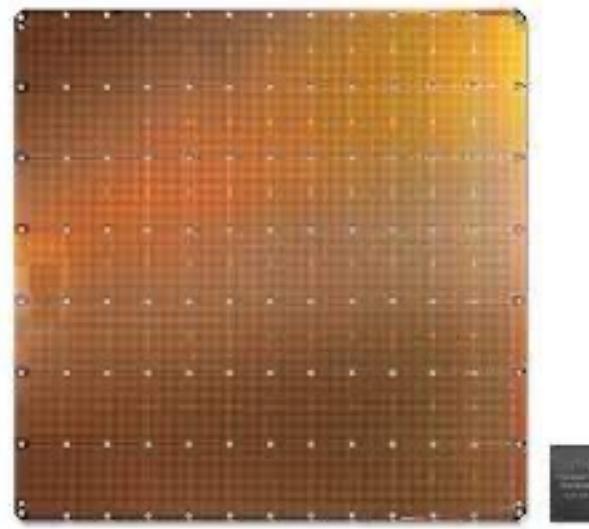
Nvidia A100



AWS's Inferentia



AMD Radeon



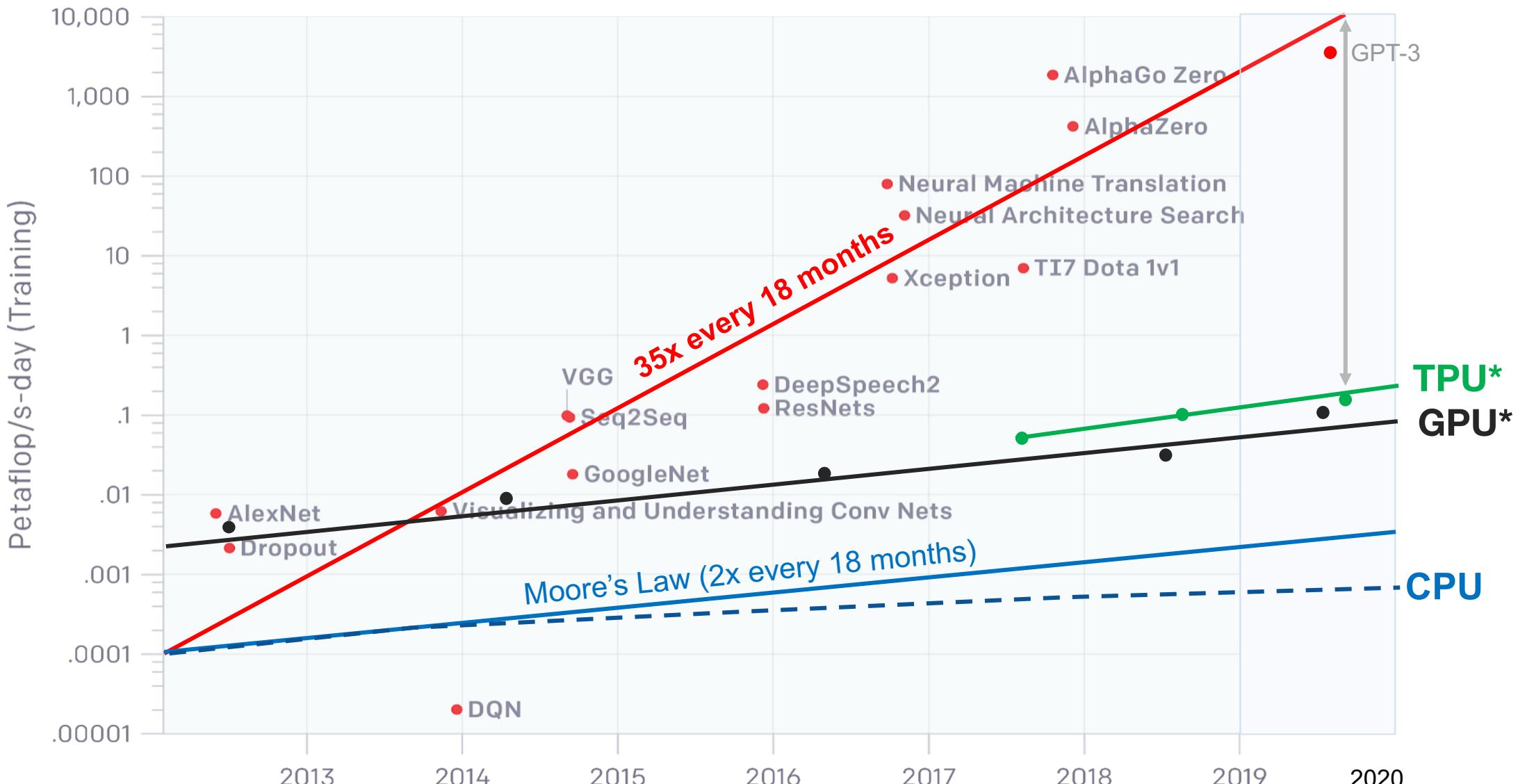
Cerebras



Intel Alchemist

Specialized hardware note enough

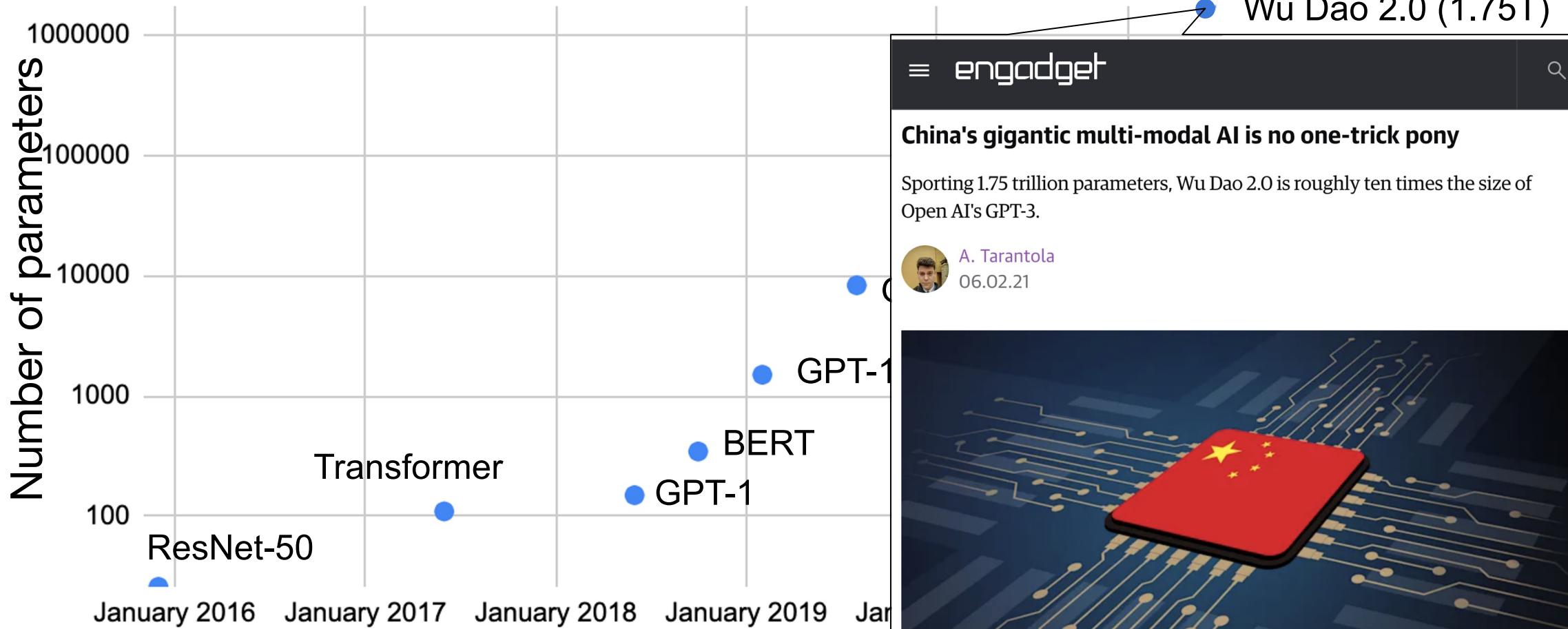
AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



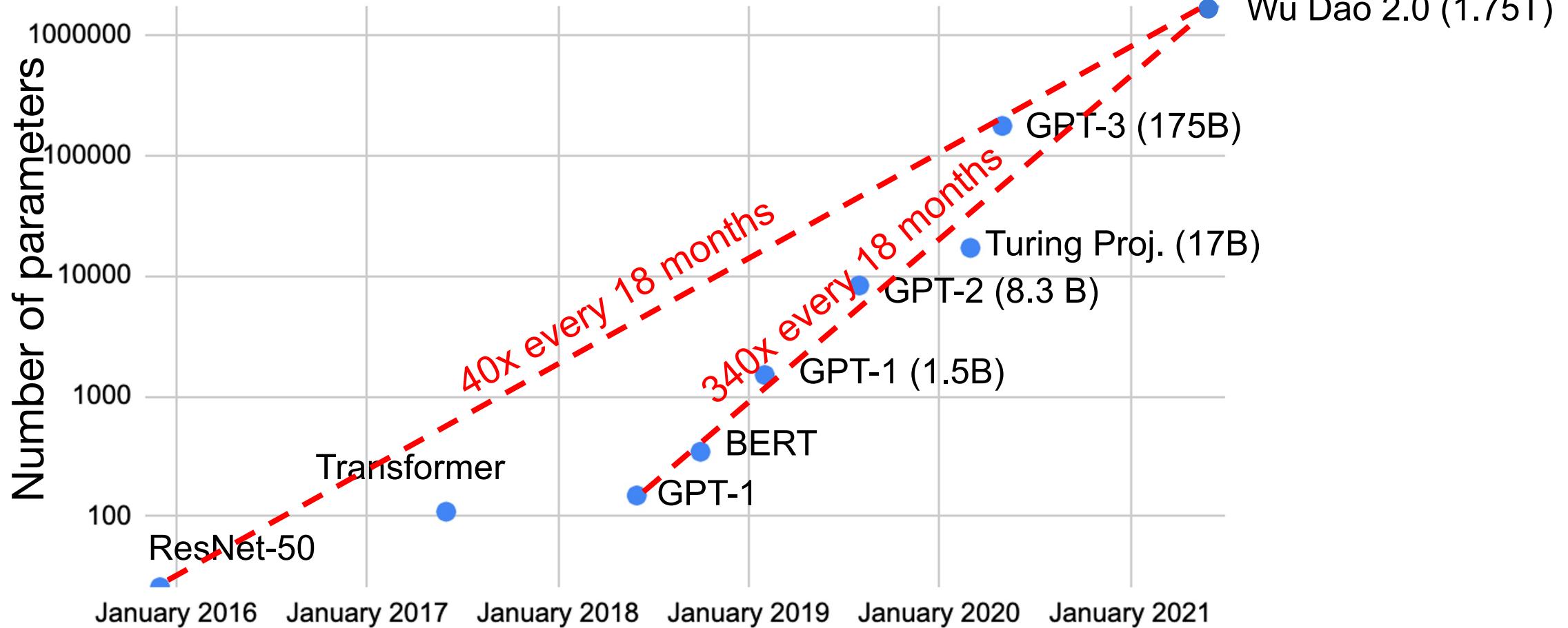
* assume 0.33 utilization

(<https://openai.com/blog/ai-and-compute/>)

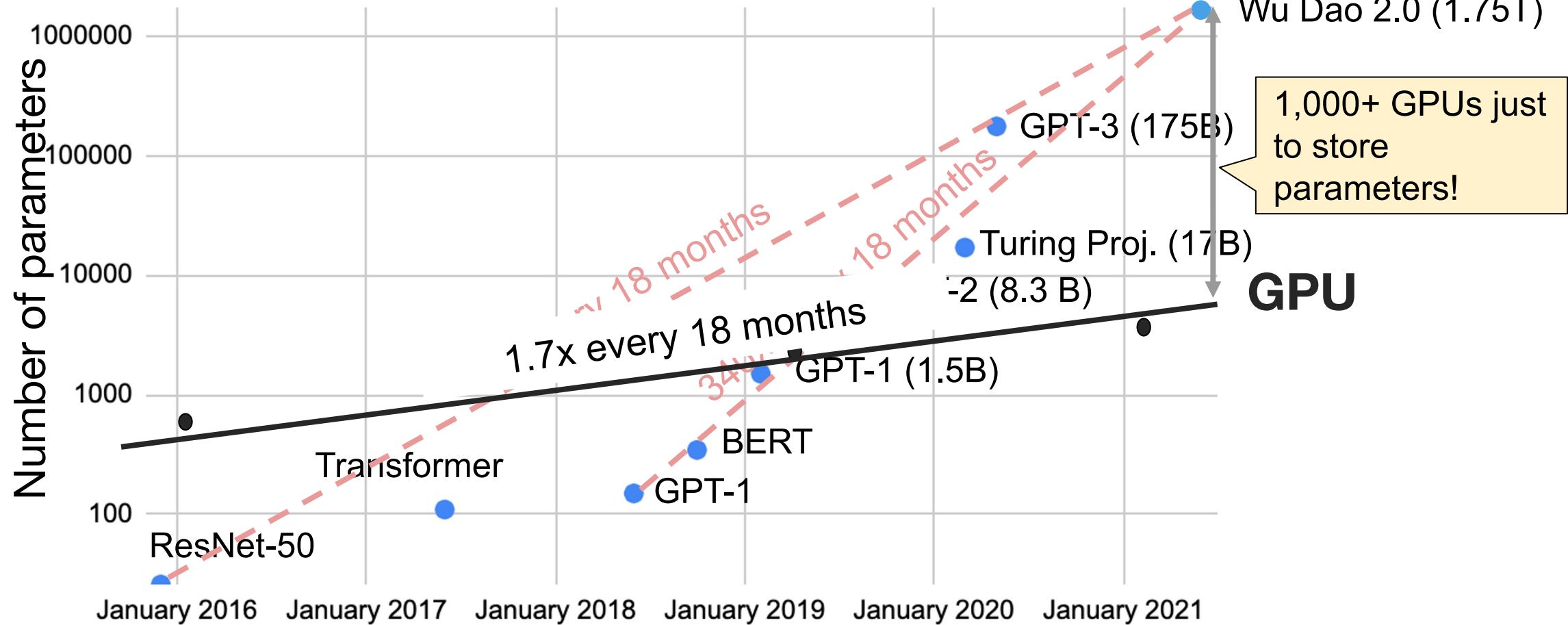
Memory demands growing as fast



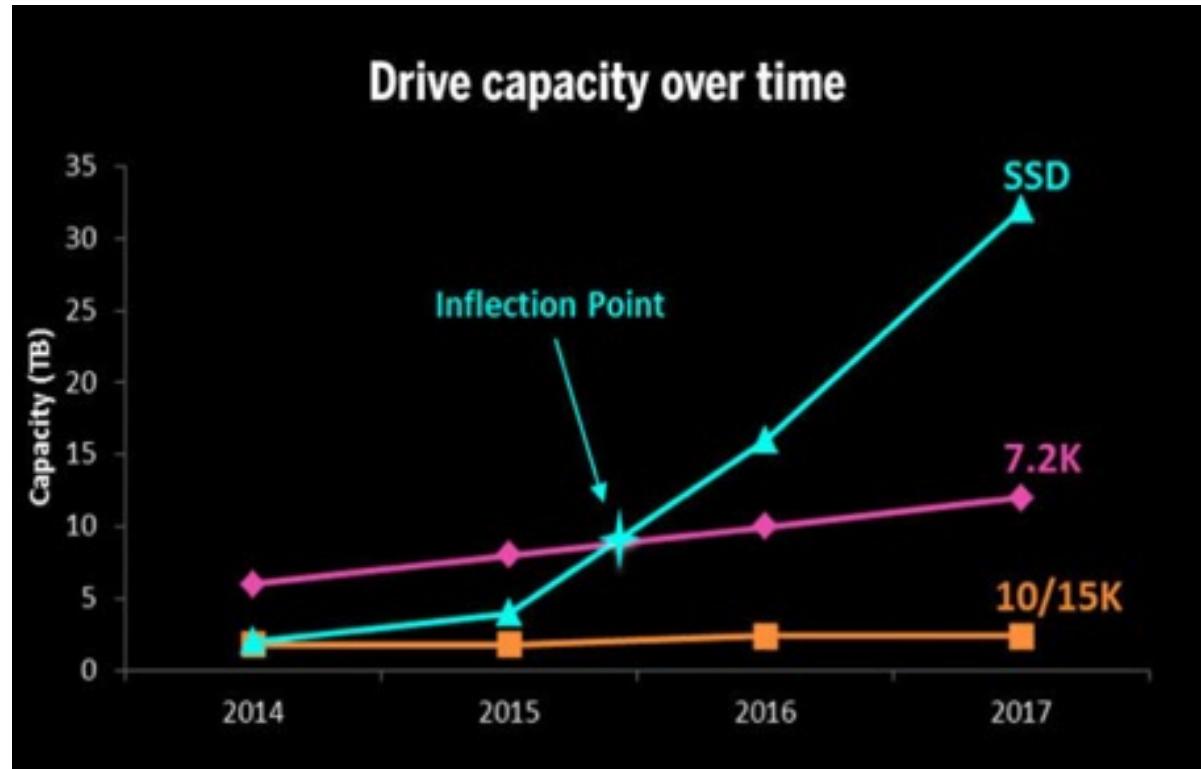
Memory demands growing as fast



Huge Gap between memory demands and single-chip memory



Storage Capacity is Still Growing!



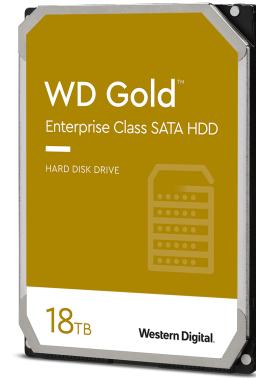
Storage Capacity



Largest SSD 3.5-inch drive:
100 TB @ \$40K (\$400/TB)



Samsung SSD 2.5-inch drive:
4 TB @ \$380K (\$95/TB)



Largest HDD 3.5-inch drive:
18 TB @ \$600 (\$33/TB)

Cheaper, but

- Slower (10x-100x)
- Consumes more power
- Less reliable

SSD/Flash will Dominate

Wikibon Projection of HDD, Tape , Other Flash, & SSD Annual Shipments, 2020-2030 (Exabytes)

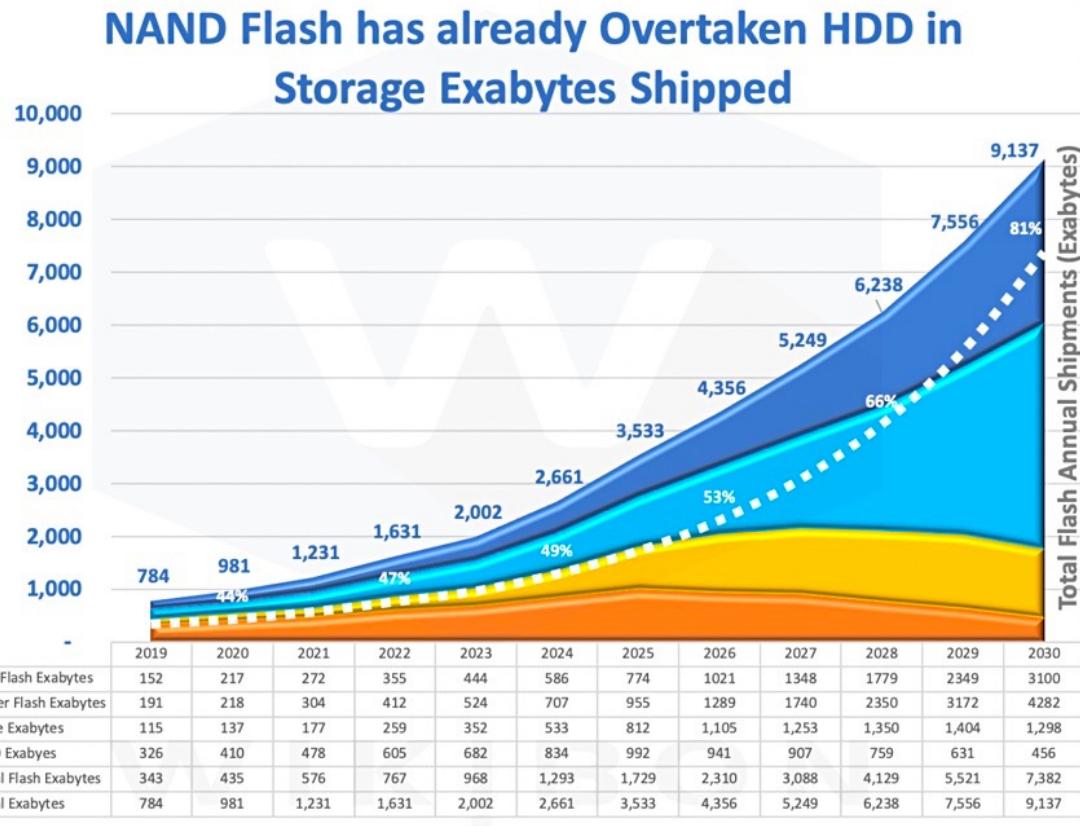


Figure 9 - Exabyte Storage Shipments Split by SSD, Other Flash, HDDs, and Tape
Source: © Wikibon, 2021.

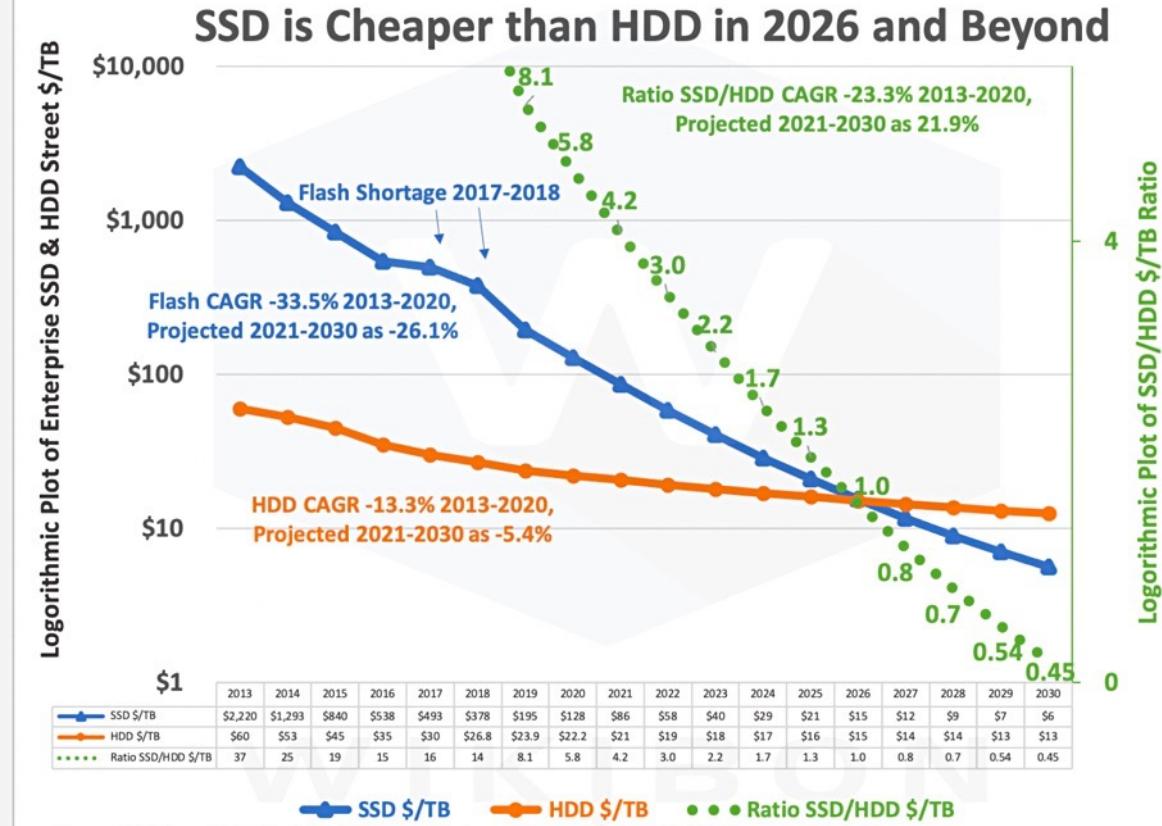
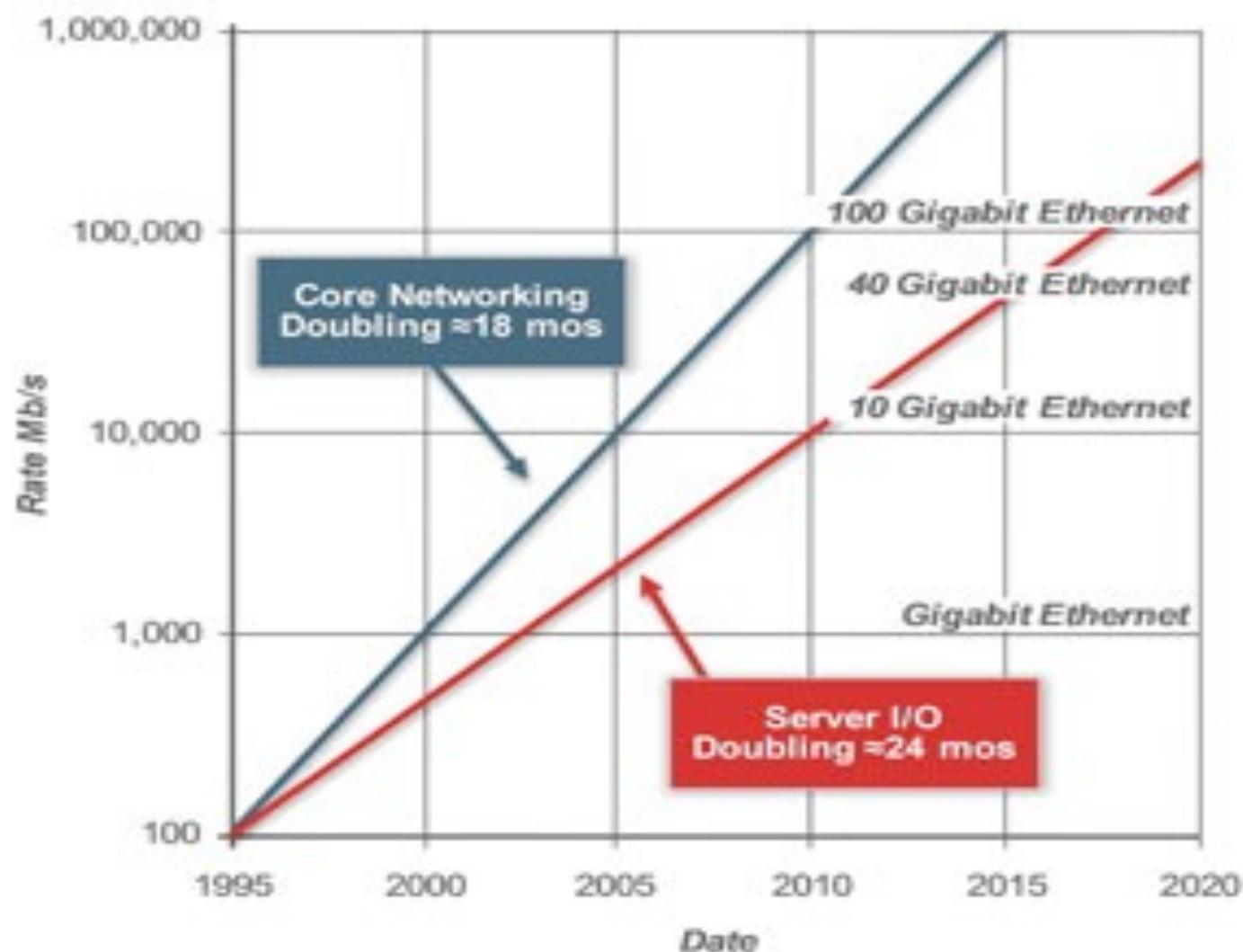


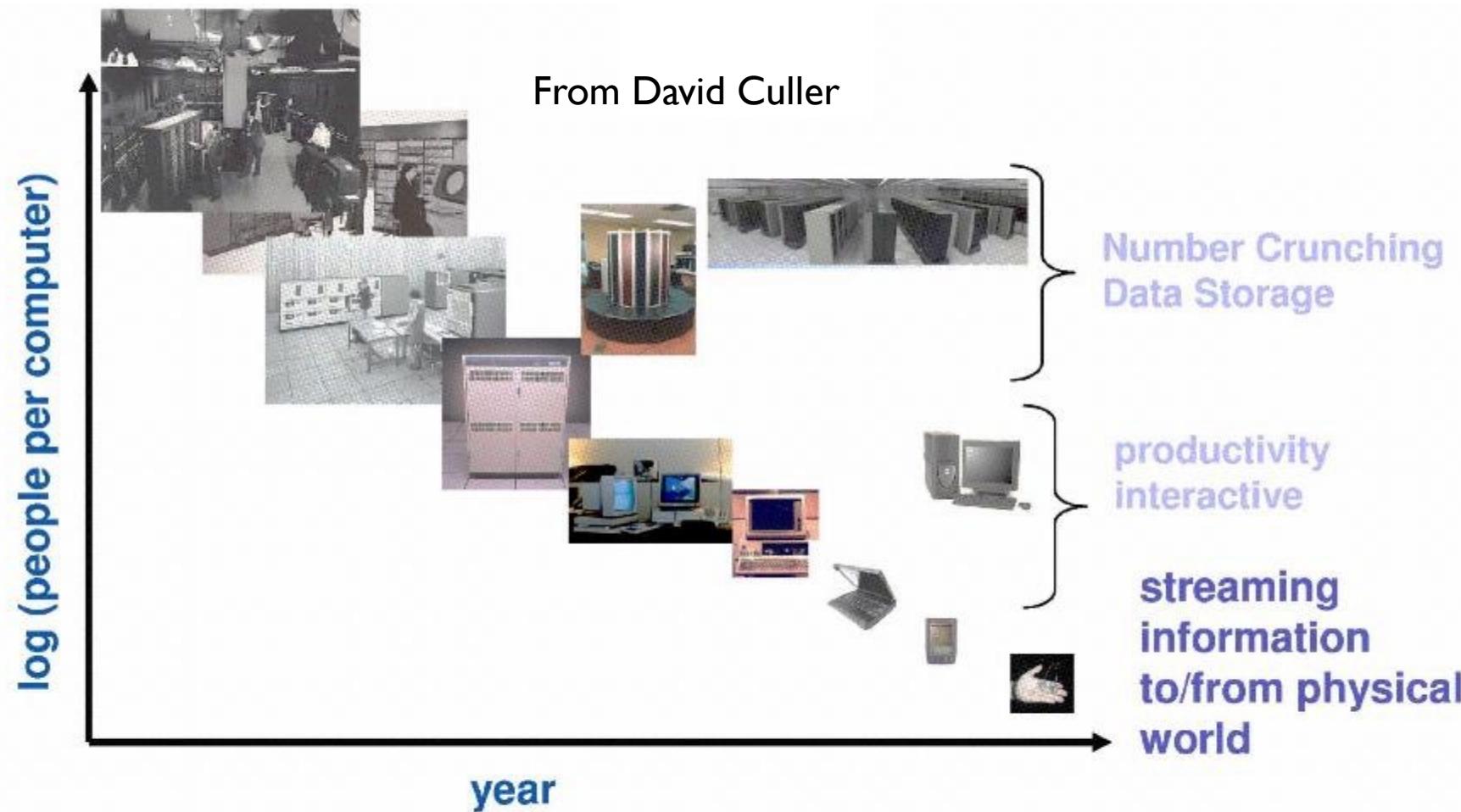
Figure 4 - SSD/HDD Pricing Ratio 2013 - 2030
Source: © Wikibon, 2021.

Network Capacity



(source: <http://www.ospmag.com/issue/article/Time-Is-Not-Always-On-Our-Side>)

People-to-Computer Ratio Over Time



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

And Range of Timescales

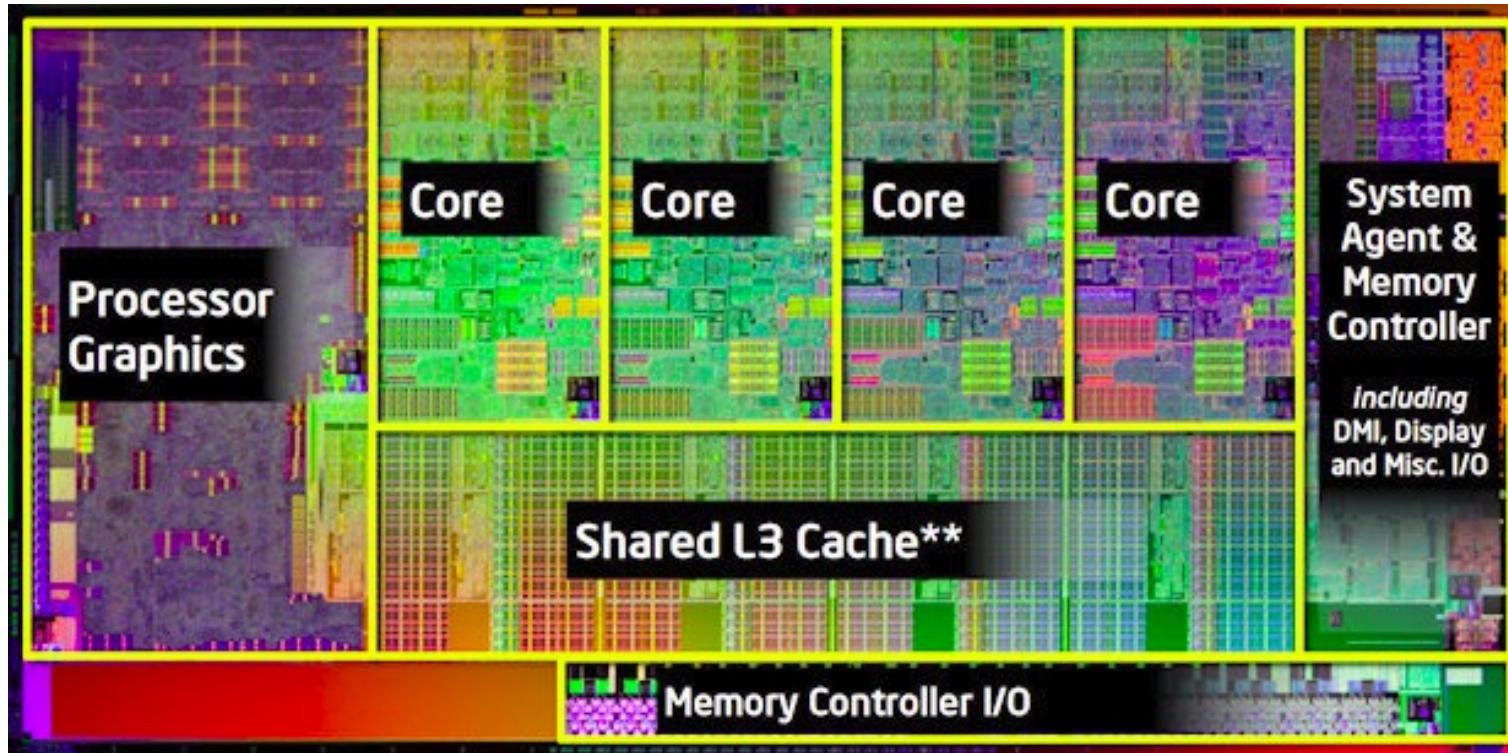
Jeff Dean:“Numbers Everyone Should Know”

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

Challenge: 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
 - The question is not whether there are bugs but how serious are the bugs!

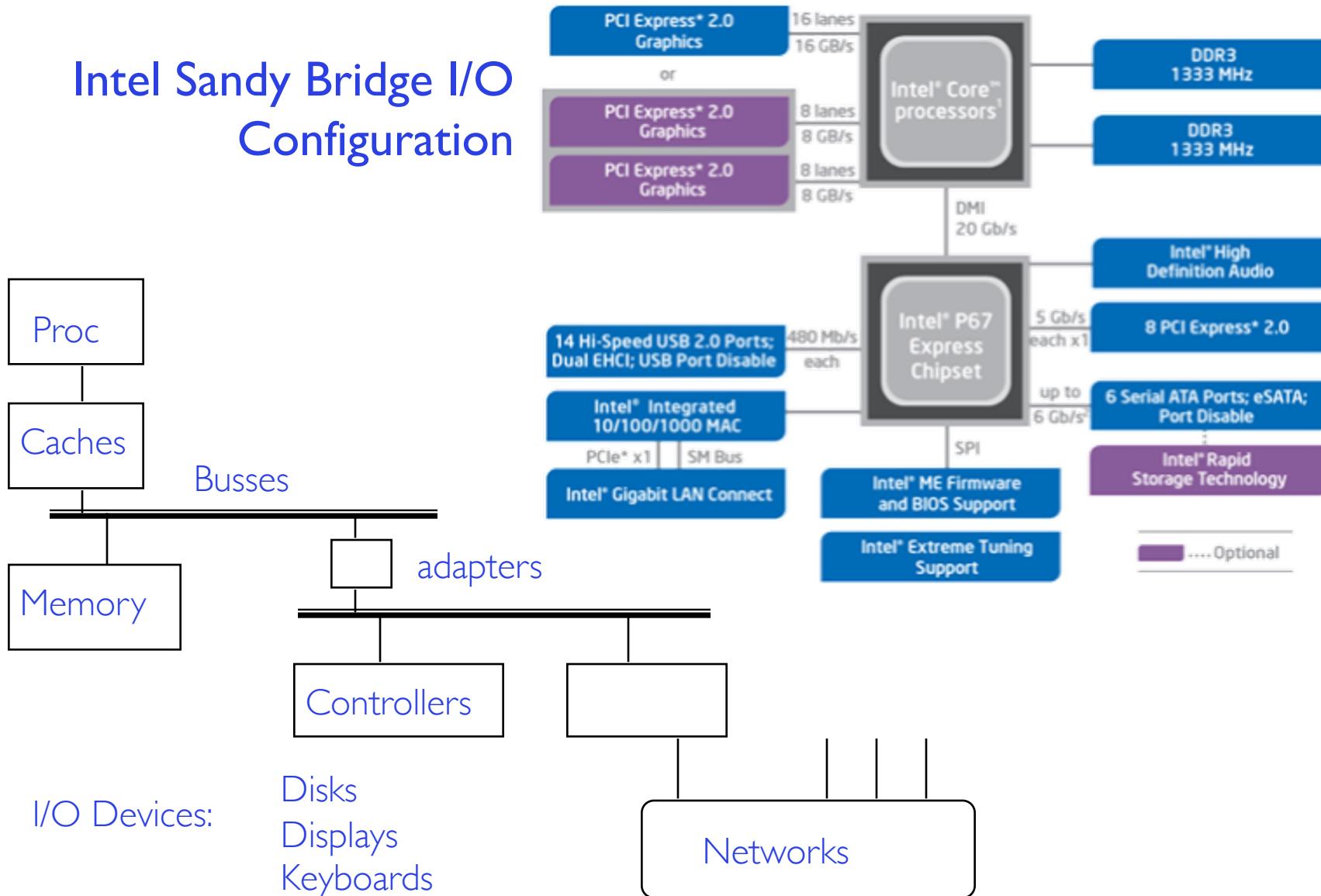
A Modern Processor: Intel Sandy Bridge



- 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
- 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

HW Functionality comes with great complexity!

Intel Sandy Bridge I/O Configuration

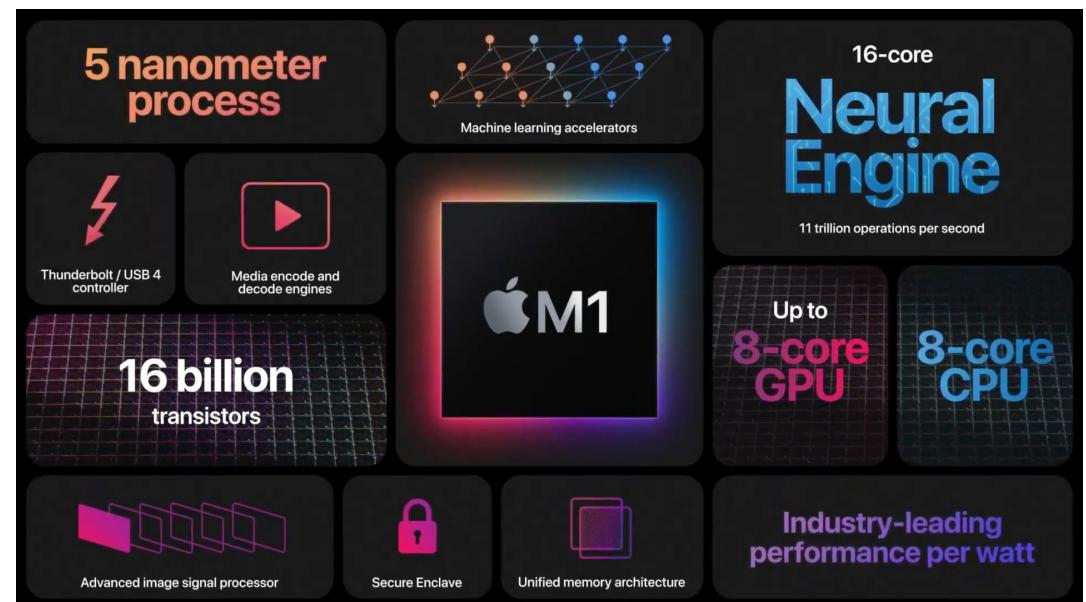
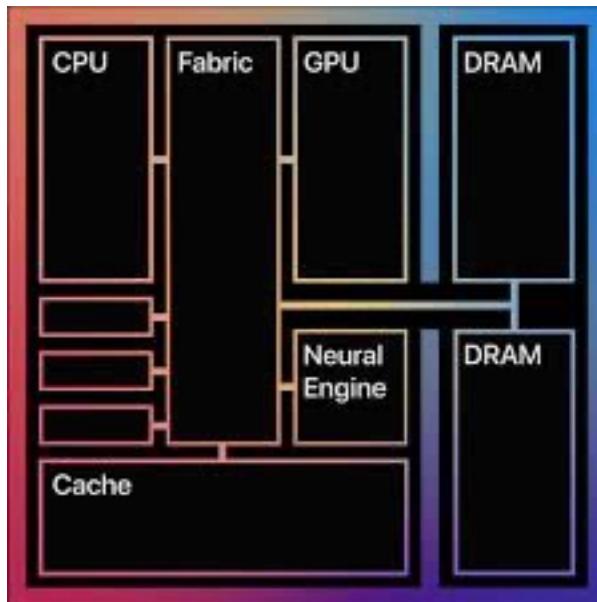


Not only specialized processors...

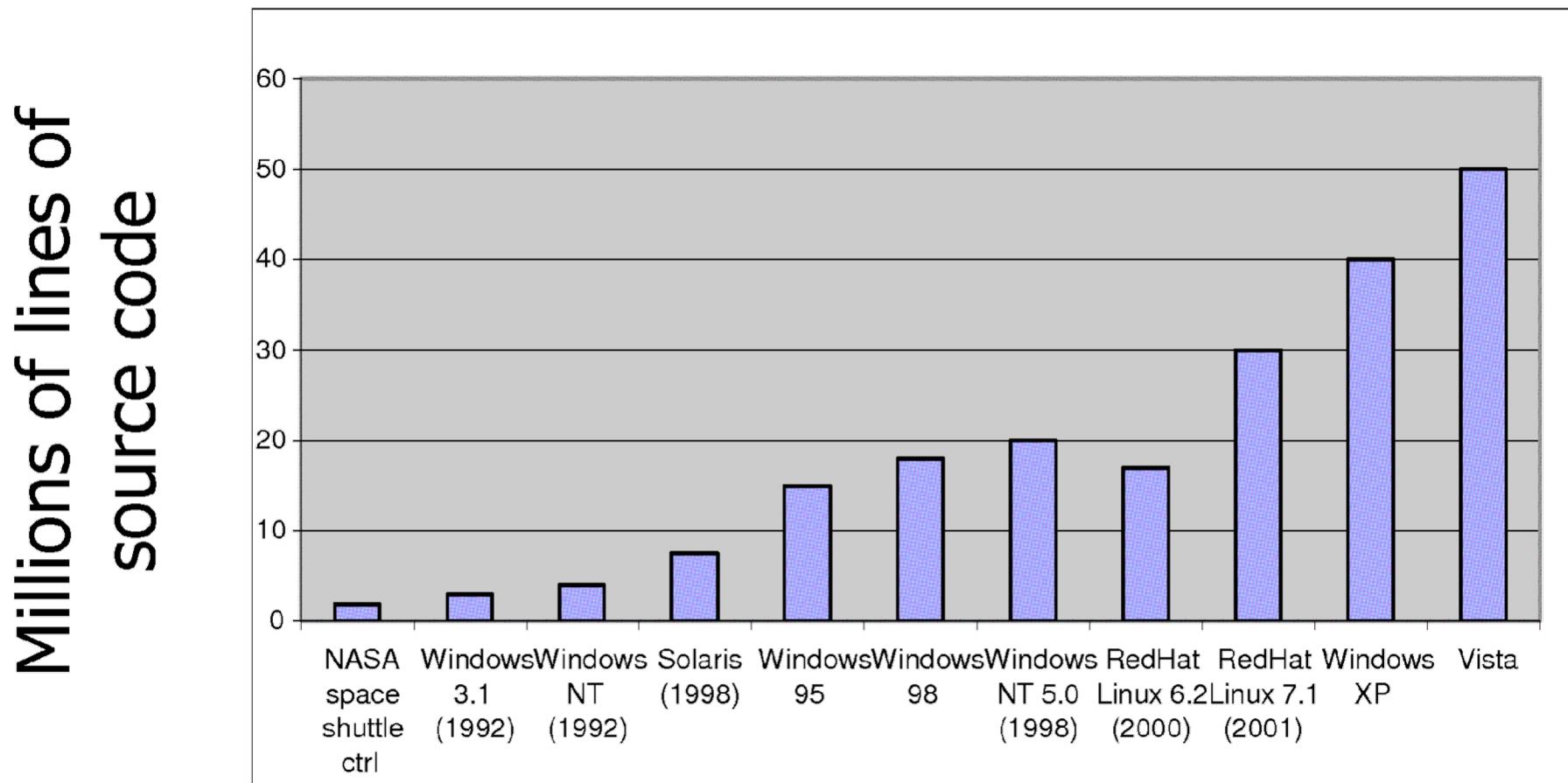
- Multi-core processors
 - Intel Xeon Phi: 64 cores
 - AMD Epyc: 64 cores; planned 128 cores (?)



- System On a Chip (SOC): accelerating trend



Increasing Software Complexity

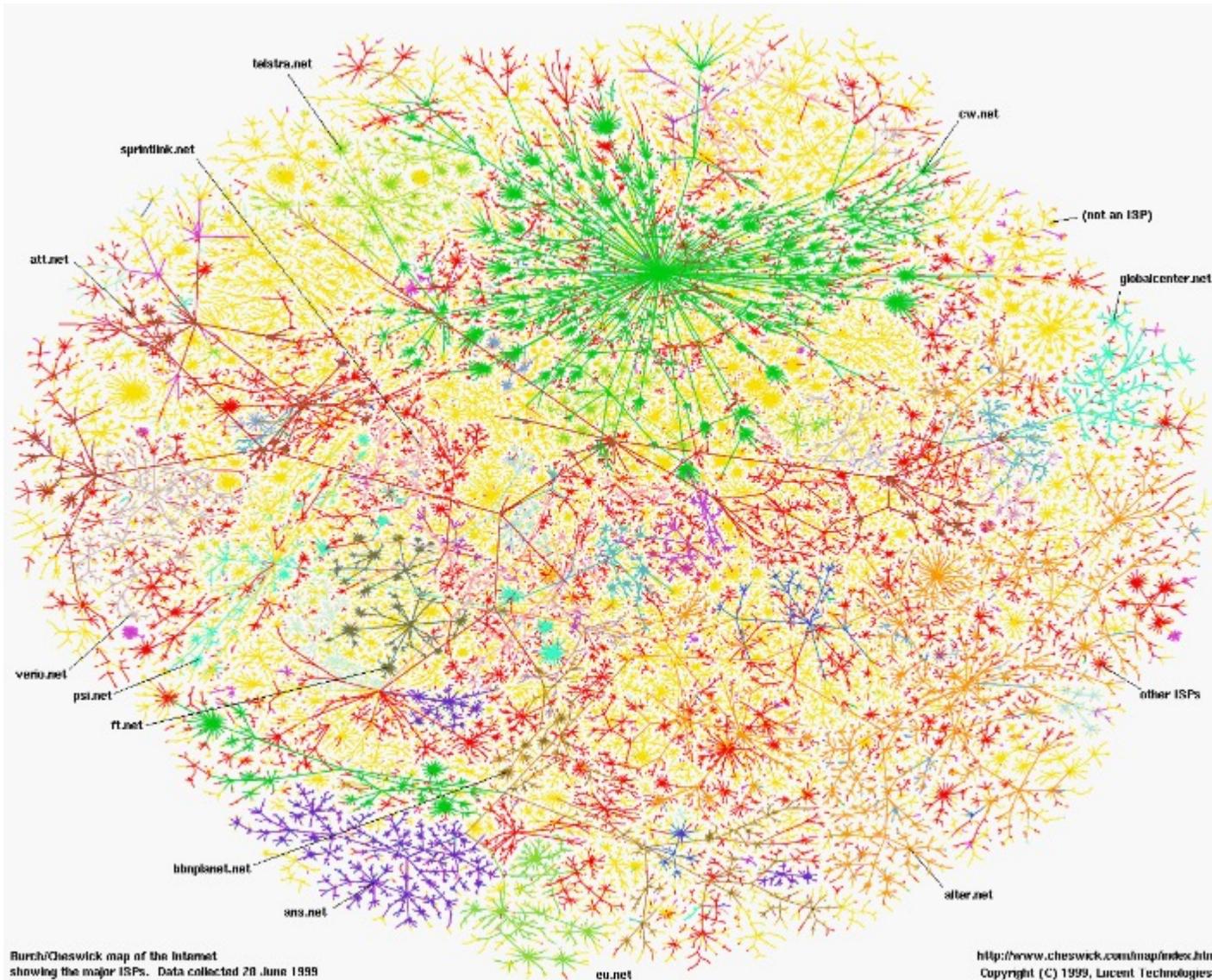


From MIT's 6.033 course

Everything going distributed

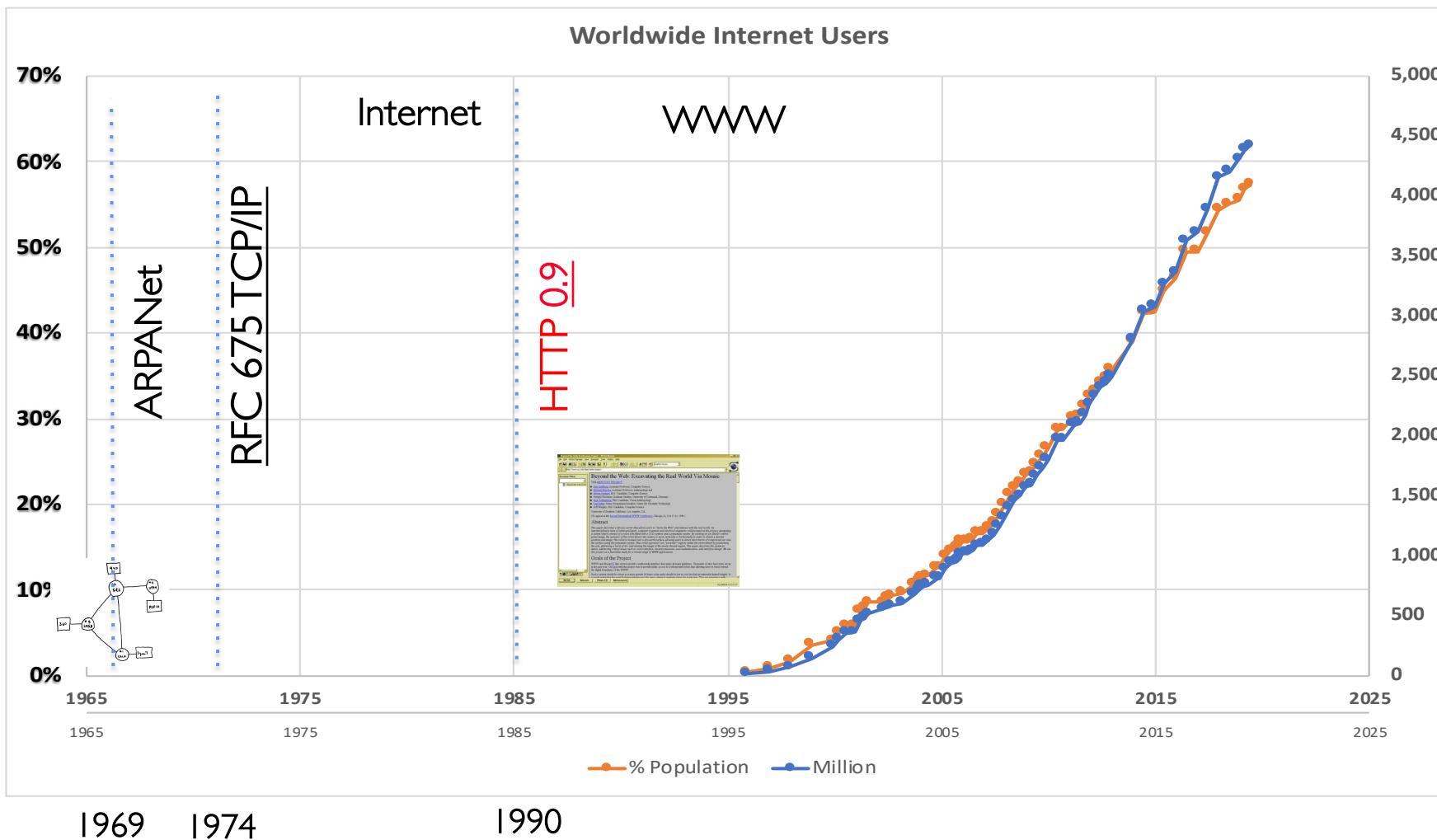
- Need to scale
 - Machine learning workloads
 - Big Data analytics
 - Scientific computing
 - ...
- Everything is connected!

Greatest Artifact of Human Civilization...



Map of the Internet

Running Systems at Internet Scale

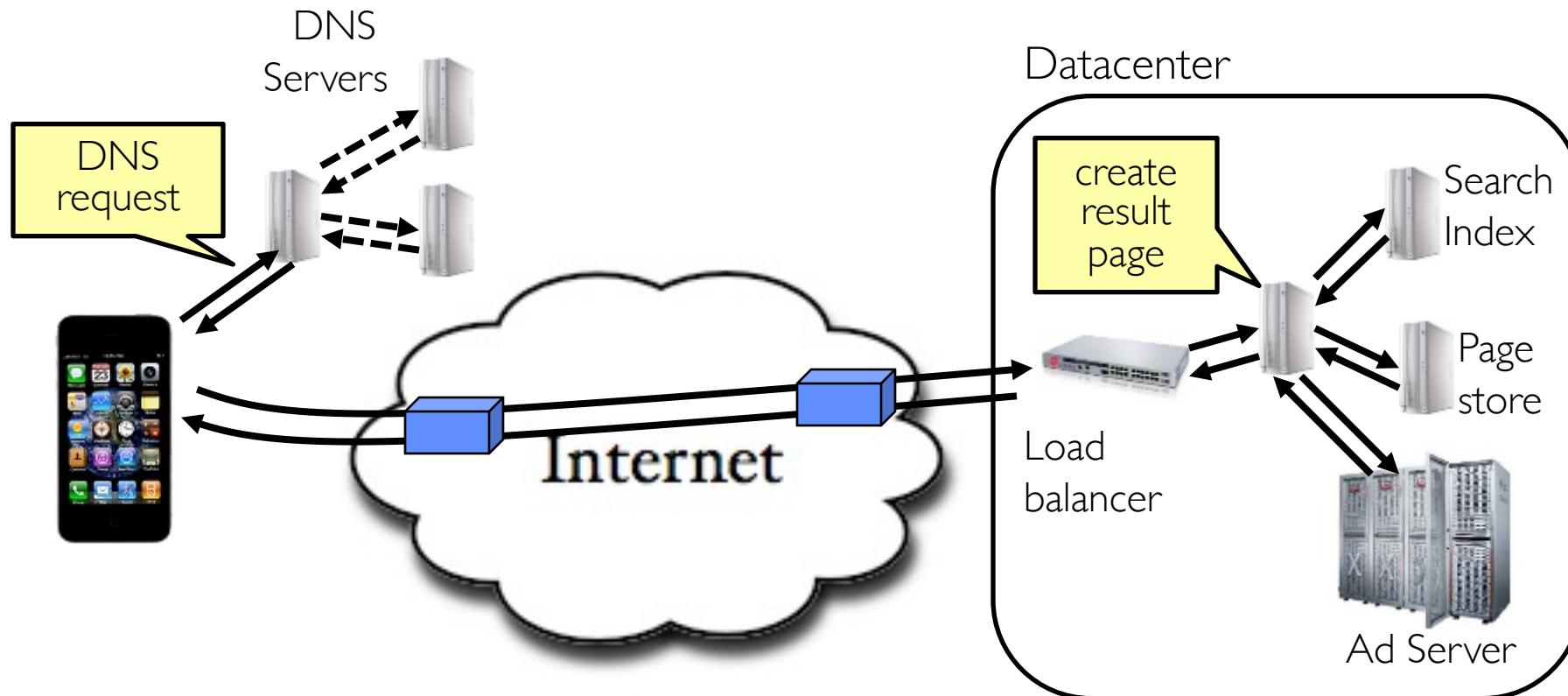


Societal Scale Information Systems (Or the “Internet of Things”?)

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



Example: What's in a Search Query?



- Complex interaction of multiple components in multiple administrative domains
 - Systems, services, protocols, ...

How do we tame complexity?

- Every piece of computer hardware different
 - Different CPU
 - » x86 (Intel, AMD), ARM, RISC-V
 - Different specialized hardware
 - » Nvidia GPUs, TPUs, AWS Inferentia, ...
 - Different amounts of memory, disk, ...
 - Different types of devices
 - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers, Face recognition
 - Different networking environment
 - » Fiber, Cable, DSL, Wireless, Firewalls,...
- Questions:
 - Does the programmer need to write a single program that performs many independent activities?
 - Does every program have to be altered for every piece of hardware?
 - Does a faulty program crash everything?
 - Does every program have access to all hardware?

Lab 0

- Booting Pintos
 - Debugging
 - Kernel Monitor
-
- Deadline: March 2 (next Thursday)

Conclusion

- Operating systems provide high-level abstractions to handle diverse hardware
 - Operating systems simplify application development by providing standard services
- Operating systems coordinate resources and protect users from each other
 - Operating systems can provide an array of fault containment, fault tolerance, and fault recovery
- Operating systems combine things from many other areas of computer science:
 - Languages, data structures, hardware, and algorithms
- Feedback: <https://www.wjx.cn/vm/t5X3TwX.aspx>