Operating Systems (Honor Track)

What is an Operating System?

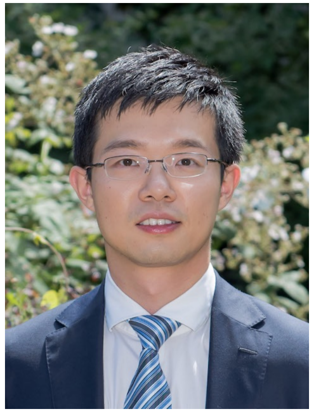
Xin Jin

Spring 2023

Acknowledgments: Ion Stoica, Berkeley CS 162

Course Staff

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

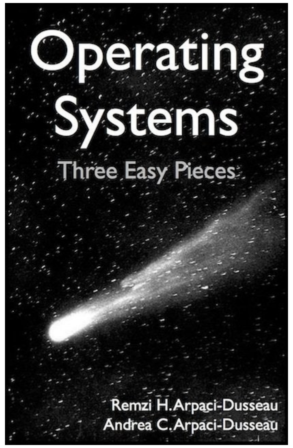


- Teaching assistants

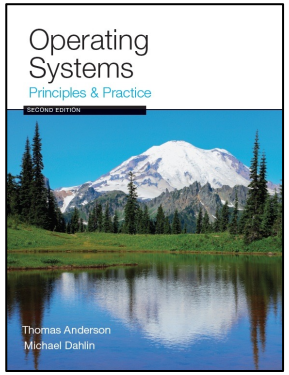
Yinmin Zhong

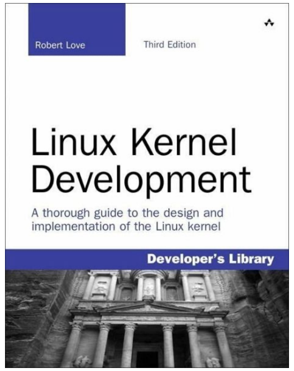
Infrastructure & Recommended Readings

- 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

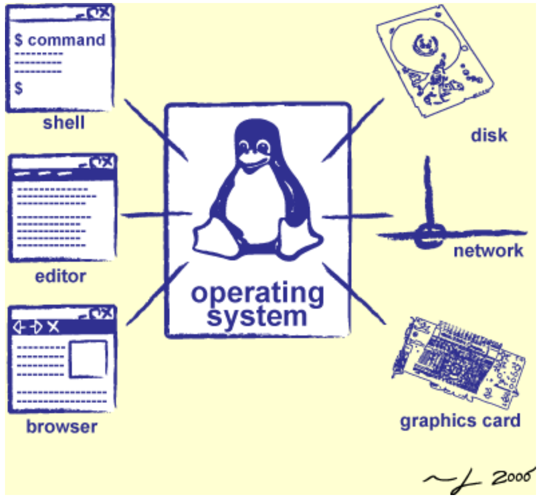
- Feedback survey: https://www.wjx.cn/vm/t5X3TwX.aspx
  - Feel free to send us your feedback about the course.
  - It is available throughput the semester.
  - It is anonymous. A private channel to talk to me.
  - Your suggestions, comments, concerns and questions are very valuable.
  - **We will summarize 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?



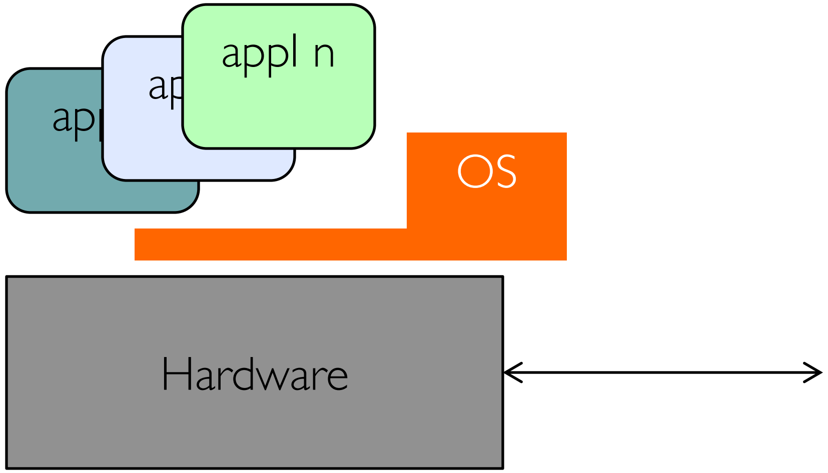
Interactive is important!

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.

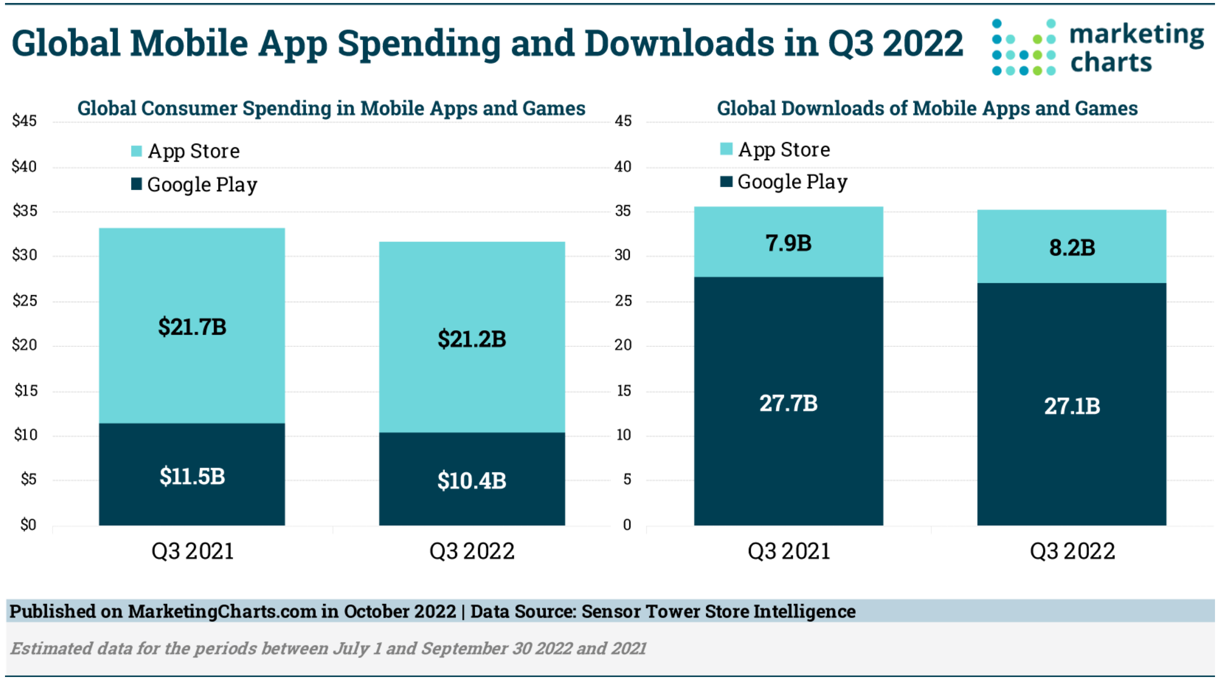
What is an operating system?

- 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



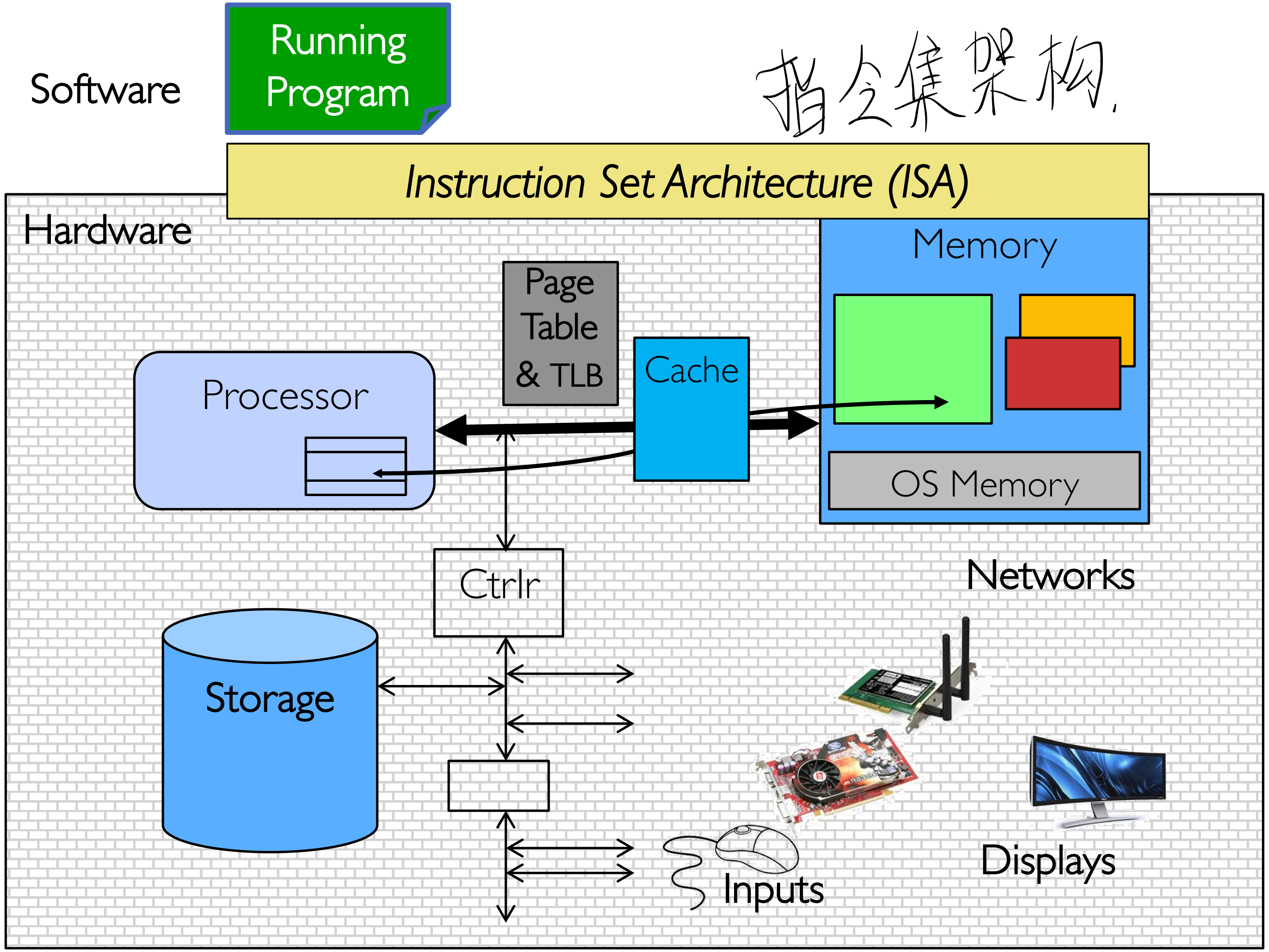
What Does an OS do?

- 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



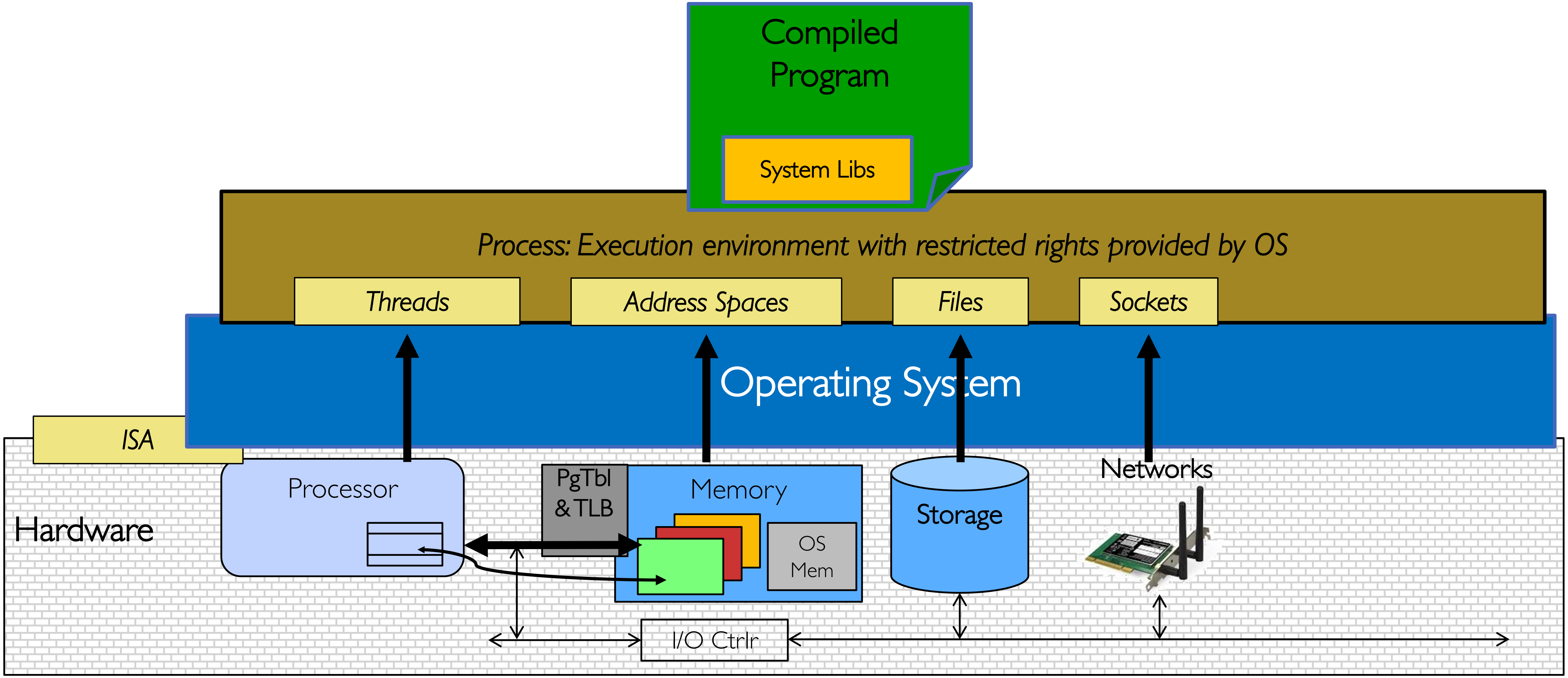
What you learned in Introduction to Computer Systems (ICS)

The OS *abstracts* these hardware details from the application

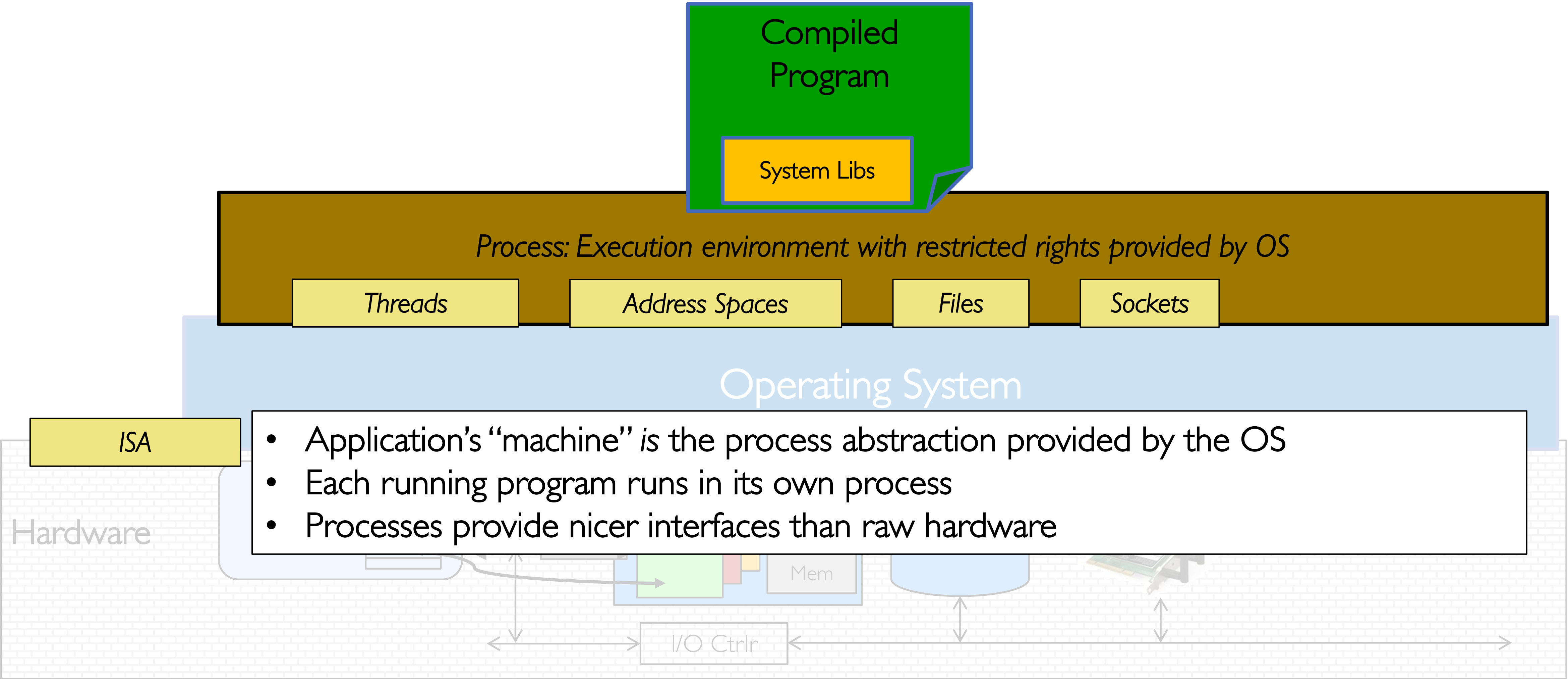
What is an Operating System?

- 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

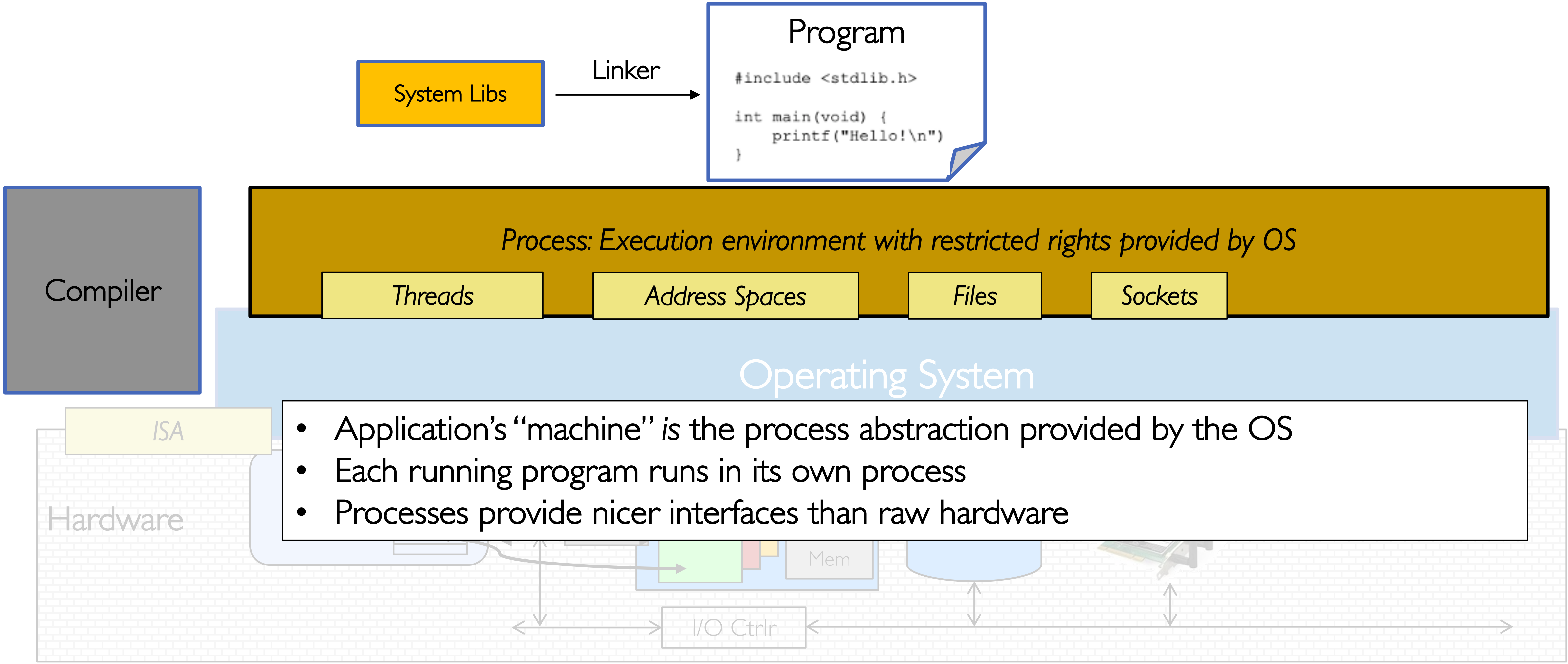


Compiled Program's View of the World



- Application's "machine" *is* the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware

System Programmer's View of the World



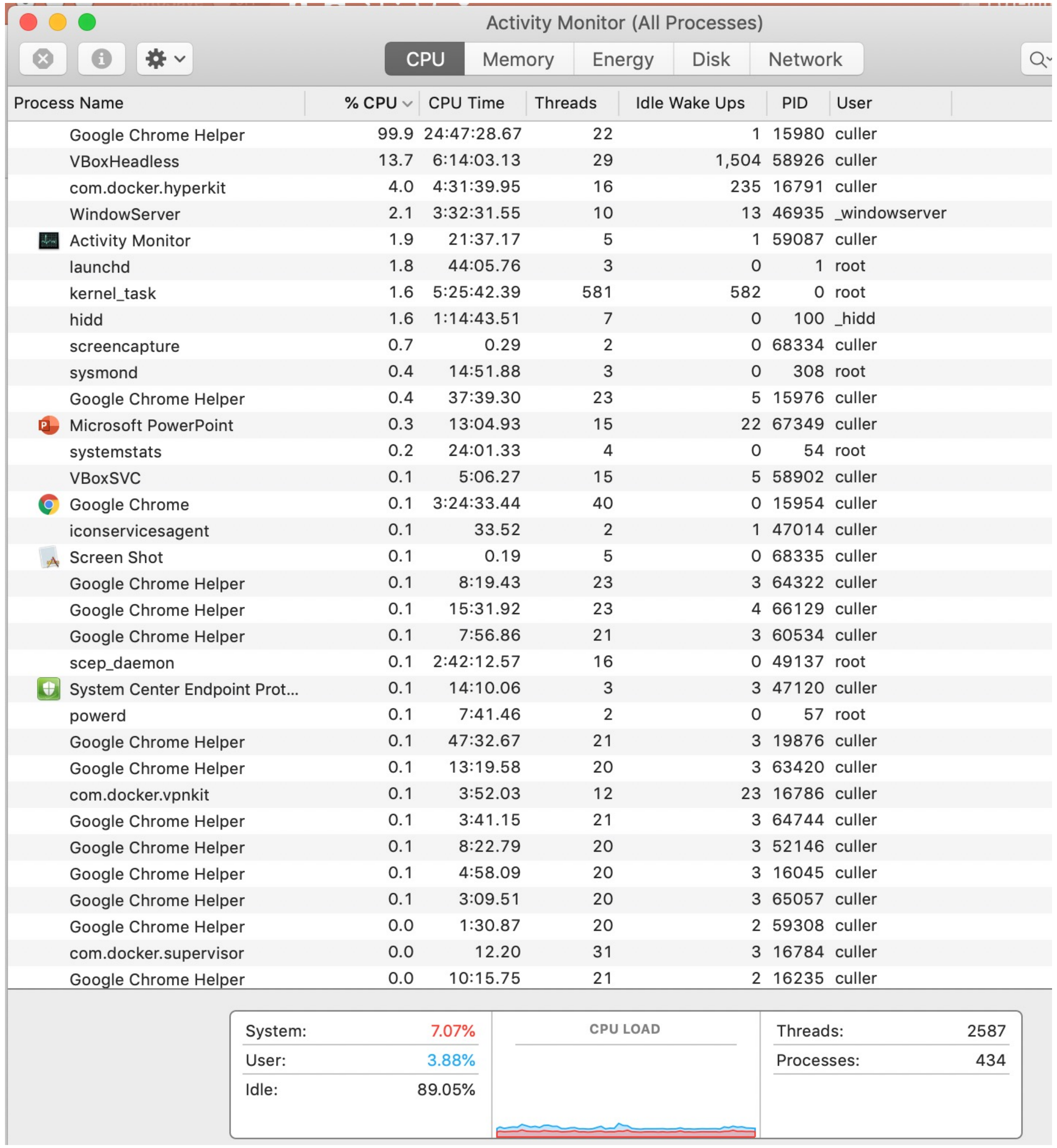
- Application's "machine" *is* the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware

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...



Activity Monitor (All Processes)

CPU | Memory | Energy | Disk | Network

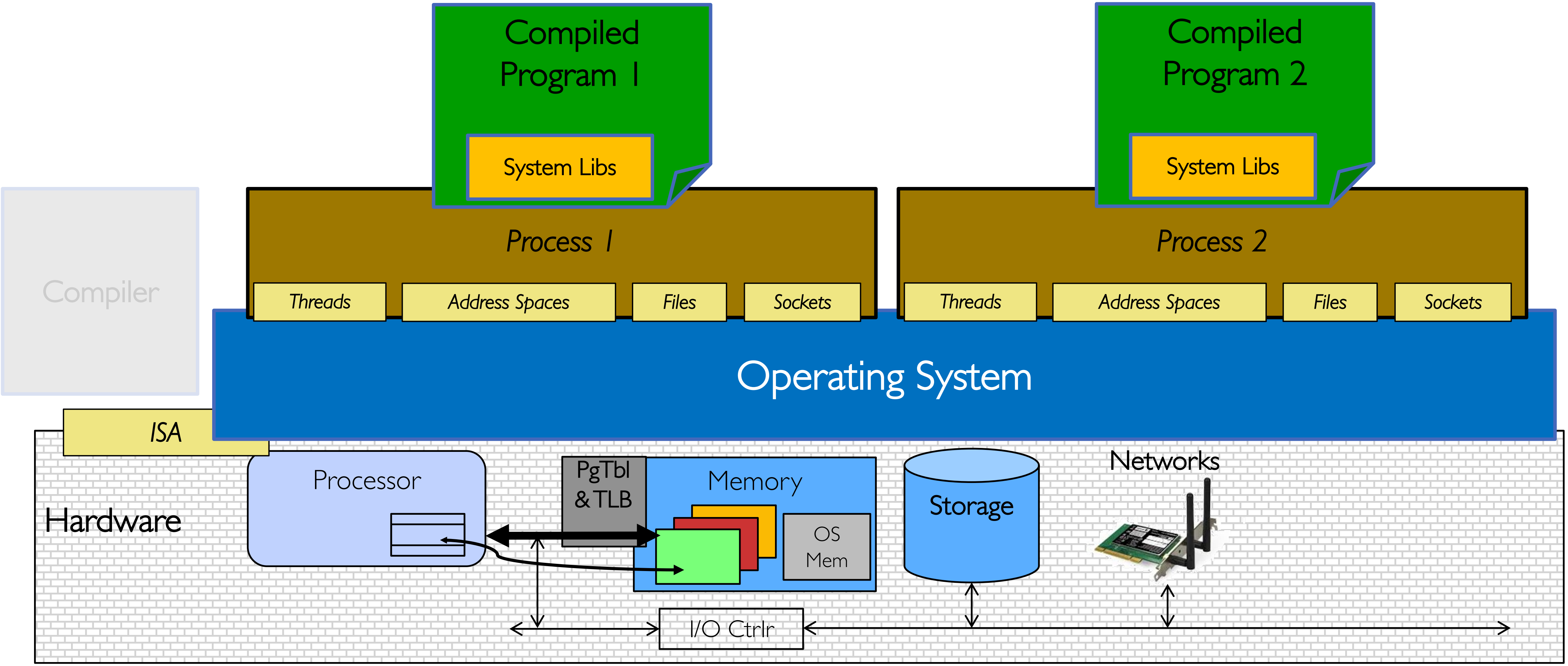
Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID	User
Google Chrome Helper	99.9	24:47:28.67	22	1	15980	culler
VBoxHeadless	13.7	6:14:03.13	29	1,504	58926	culler
com.docker.hyperkit	4.0	4:31:39.95	16	235	16791	culler
WindowServer	2.1	3:32:31.55	10	13	46935	_windowserver
Activity Monitor	1.9	21:37.17	5	1	59087	culler
launchd	1.8	44:05.76	3	0	1	root
kernel_task	1.6	5:25:42.39	581	582	0	root
hidd	1.6	1:14:43.51	7	0	100	_hidd
screencapture	0.7	0.29	2	0	68334	culler
sysmond	0.4	14:51.88	3	0	308	root
Google Chrome Helper	0.4	37:39.30	23	5	15976	culler
Microsoft PowerPoint	0.3	13:04.93	15	22	67349	culler
systemstats	0.2	24:01.33	4	0	54	root
VBoxSVC	0.1	5:06.27	15	5	58902	culler
Google Chrome	0.1	3:24:33.44	40	0	15954	culler
iconservicesagent	0.1	33.52	2	1	47014	culler
Screen Shot	0.1	0.19	5	0	68335	culler
Google Chrome Helper	0.1	8:19.43	23	3	64322	culler
Google Chrome Helper	0.1	15:31.92	23	4	66129	culler
Google Chrome Helper	0.1	7:56.86	21	3	60534	culler
scep_daemon	0.1	2:42:12.57	16	0	49137	root
System Center Endpoint Prot...	0.1	14:10.06	3	3	47120	culler
powerd	0.1	7:41.46	2	0	57	root
Google Chrome Helper	0.1	47:32.67	21	3	19876	culler
Google Chrome Helper	0.1	13:19.58	20	3	63420	culler
com.docker.vpnkit	0.1	3:52.03	12	23	16786	culler
Google Chrome Helper	0.1	3:41.15	21	3	64744	culler
Google Chrome Helper	0.1	8:22.79	20	3	52146	culler
Google Chrome Helper	0.1	4:58.09	20	3	16045	culler
Google Chrome Helper	0.1	3:09.51	20	3	65057	culler
Google Chrome Helper	0.0	1:30.87	20	2	59308	culler
com.docker.supervisor	0.0	12.20	31	3	16784	culler
Google Chrome Helper	0.0	10:15.75	21	2	16235	culler

System:	7.07%
User:	3.88%
Idle:	89.05%

CPU LOAD

Threads:	2587
Processes:	434

Operating System's View of the World

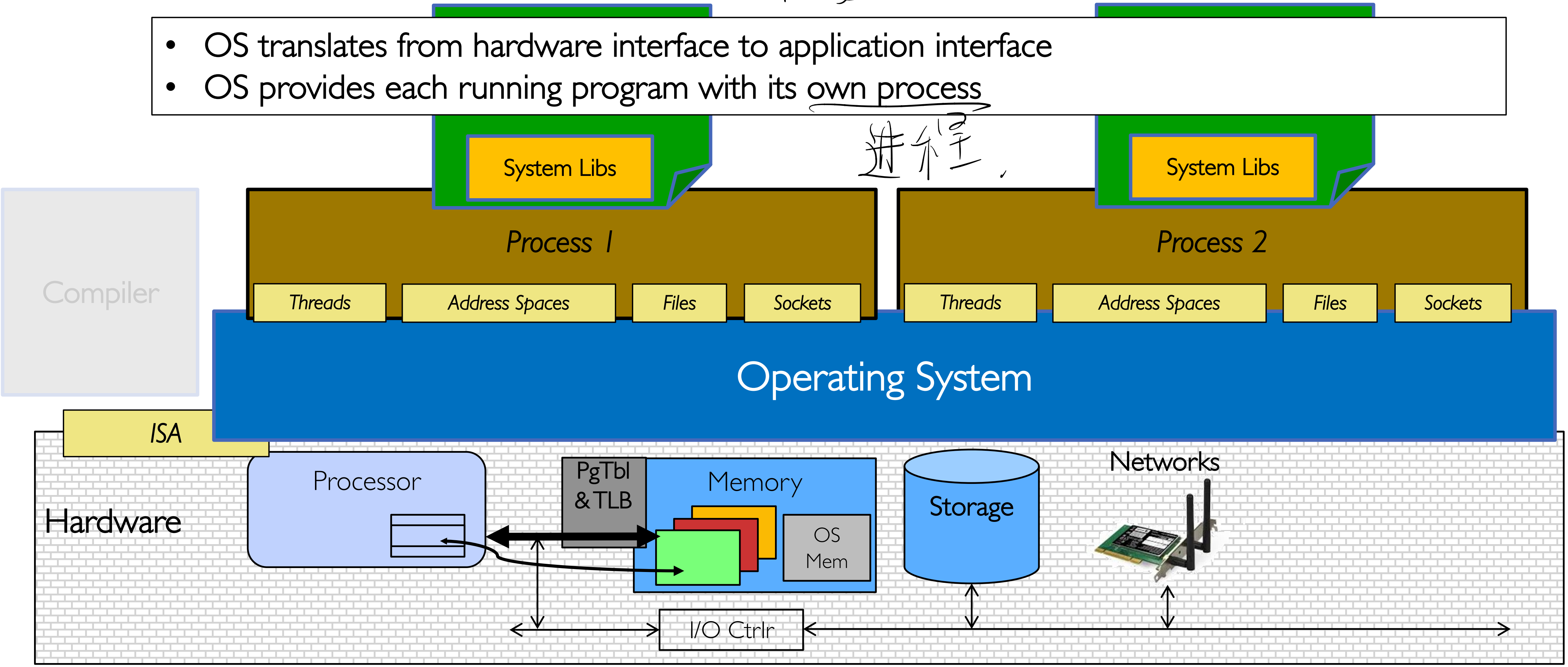


Operating System's View of the World

机器接口 → 应用接口.

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

进程.

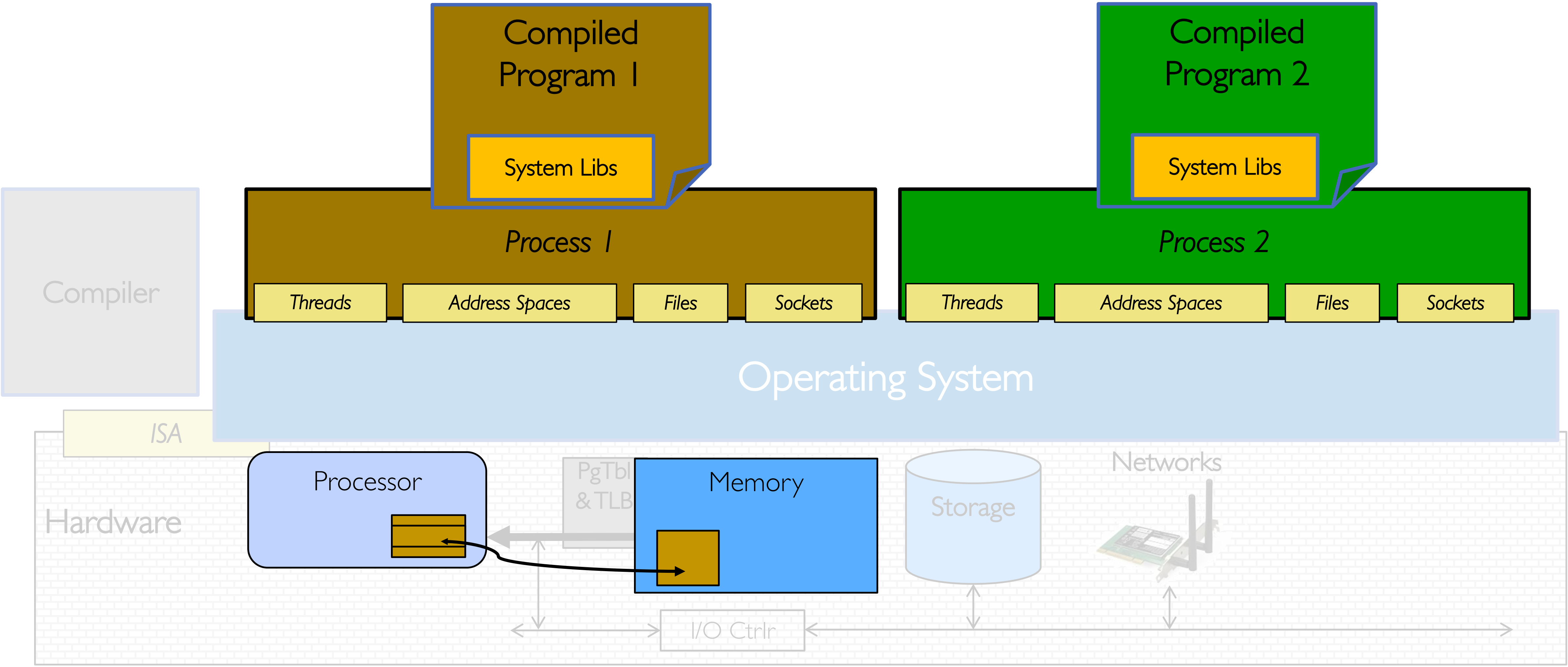


What is an Operating System?

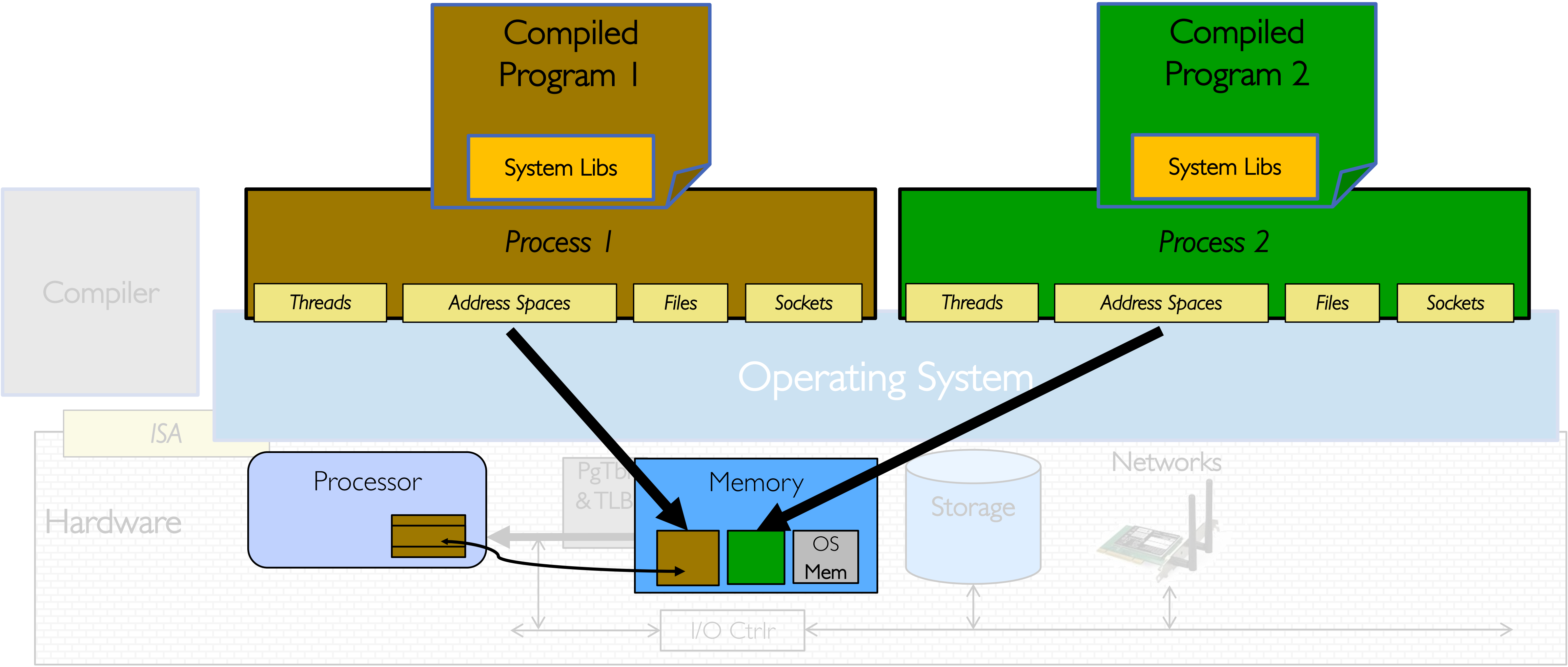
- Referee
  - Manage protection, isolation, and sharing of resources
    - » Resource allocation and communication

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

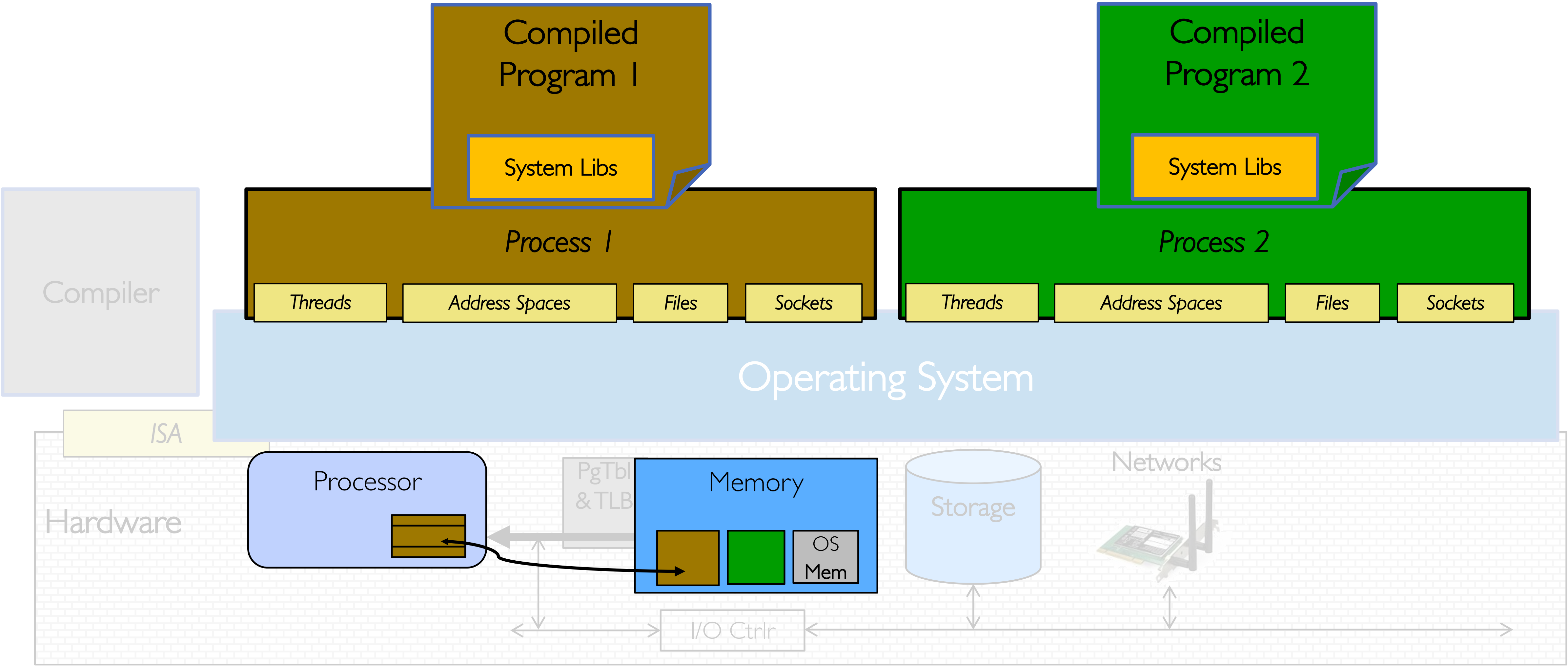
OS Basics: Running a Process



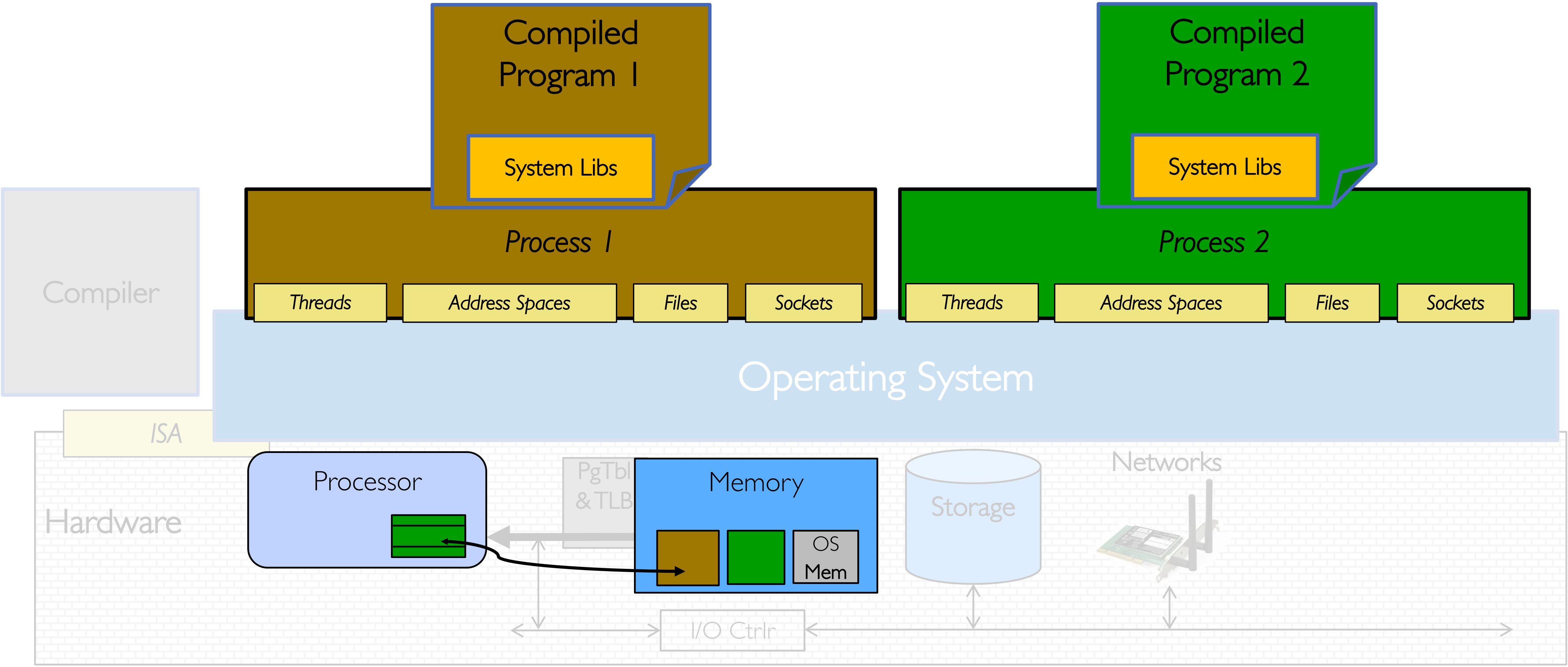
OS Basics: Switching Processes



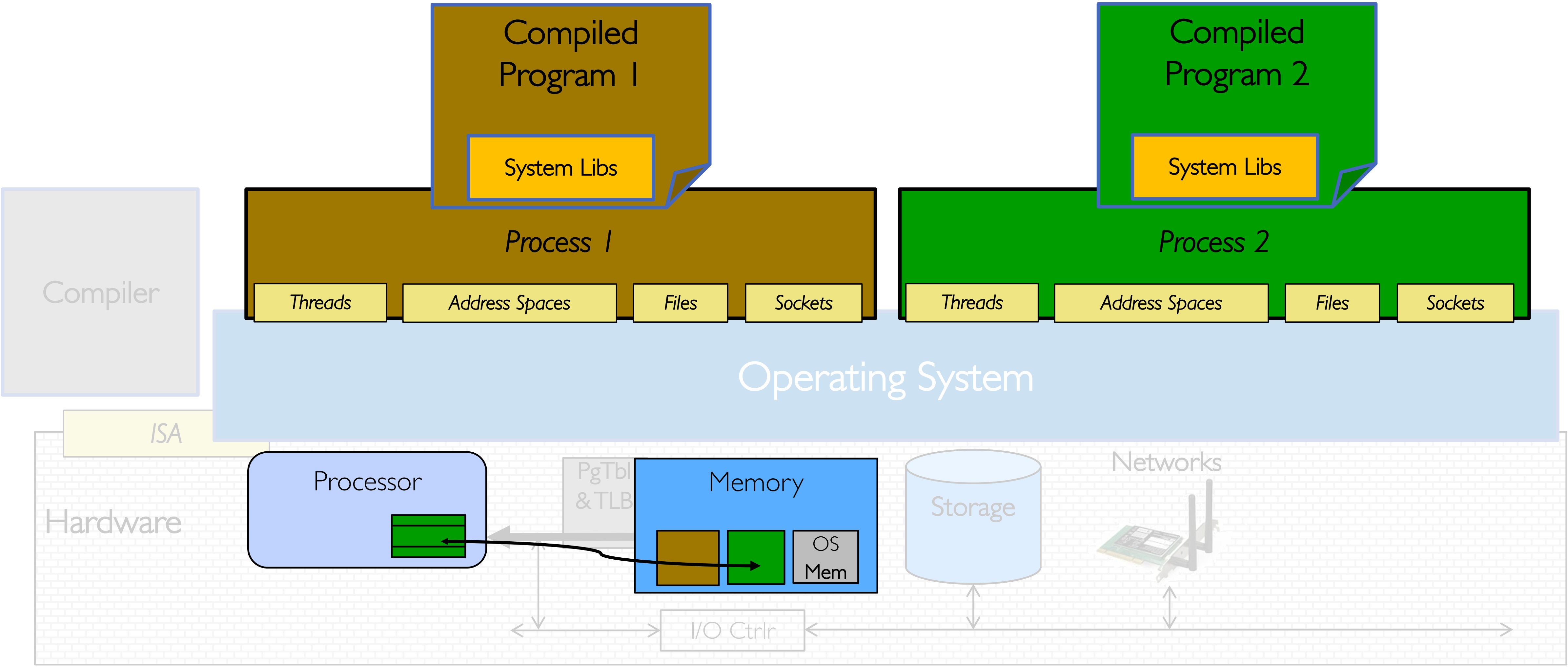
OS Basics: Switching Processes



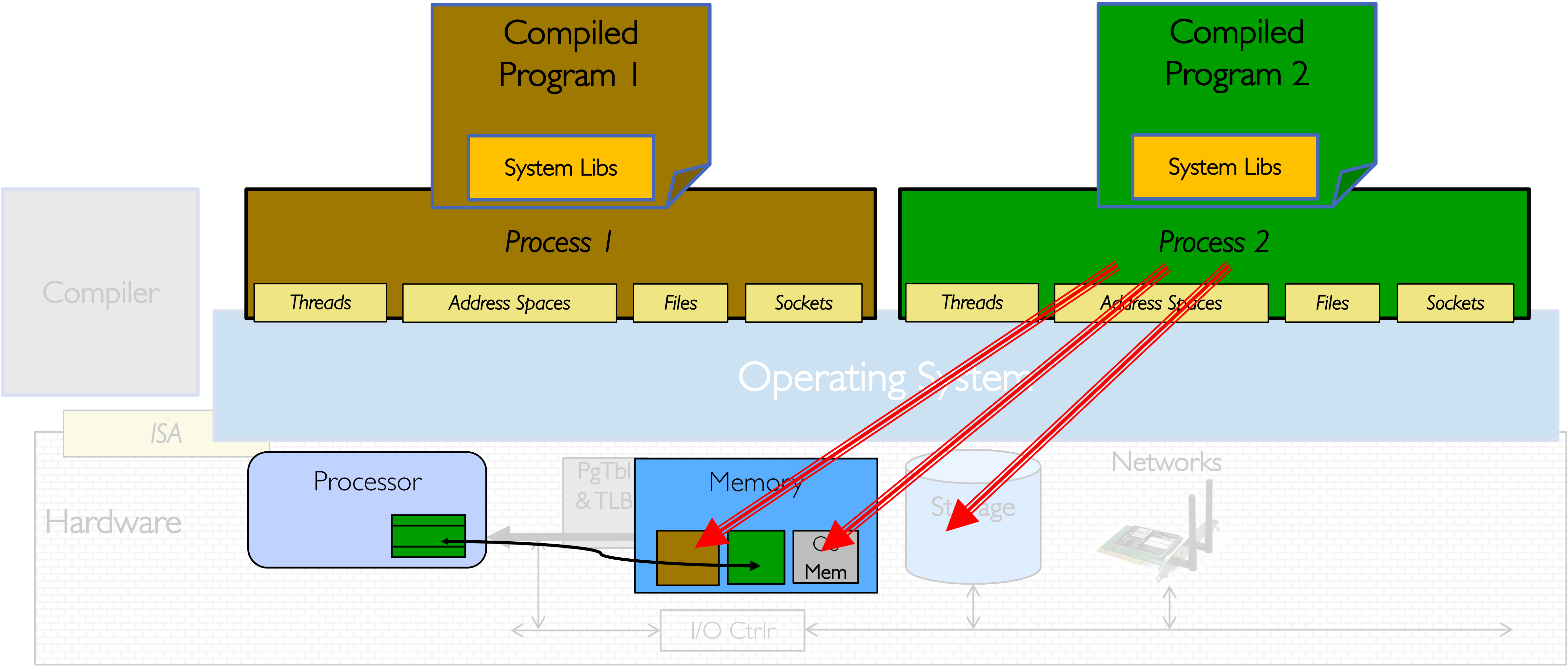
OS Basics: Switching Processes



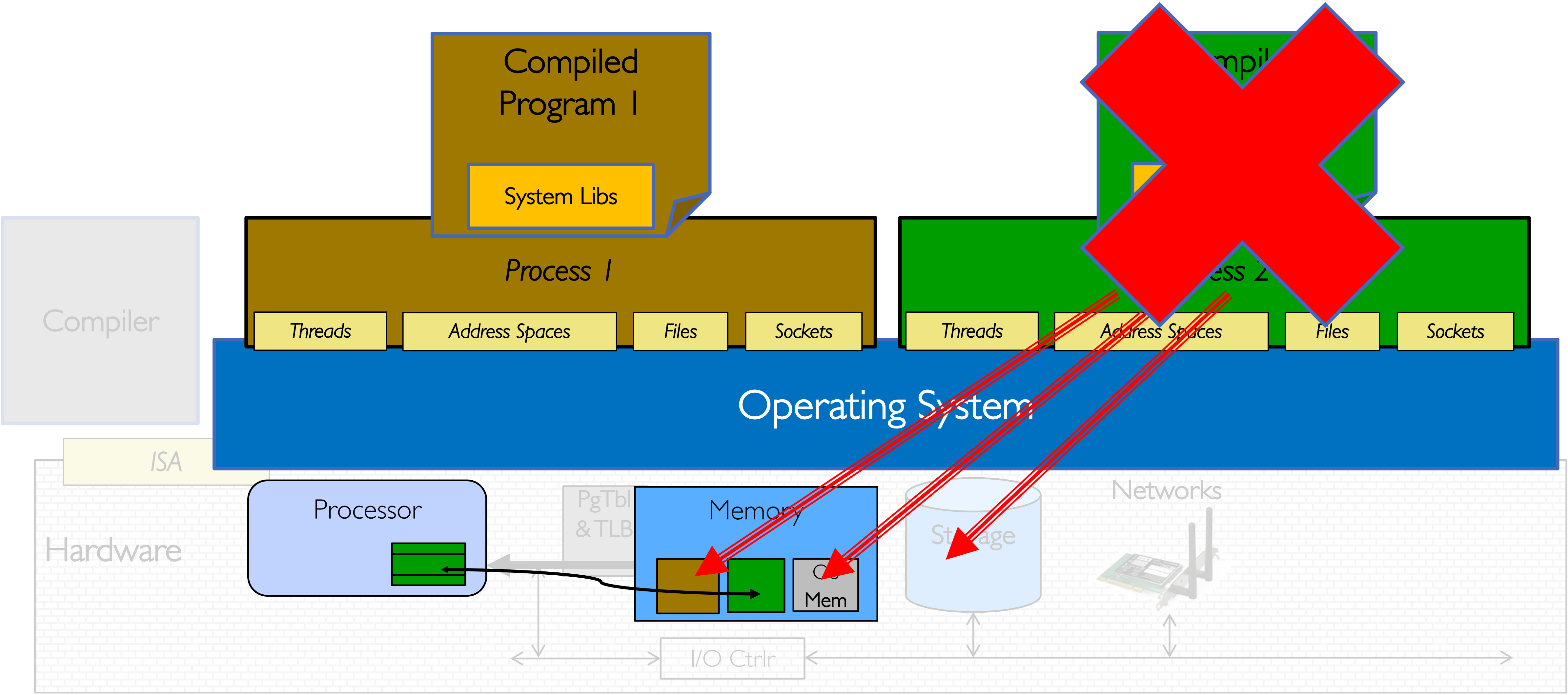
OS Basics: Switching Processes



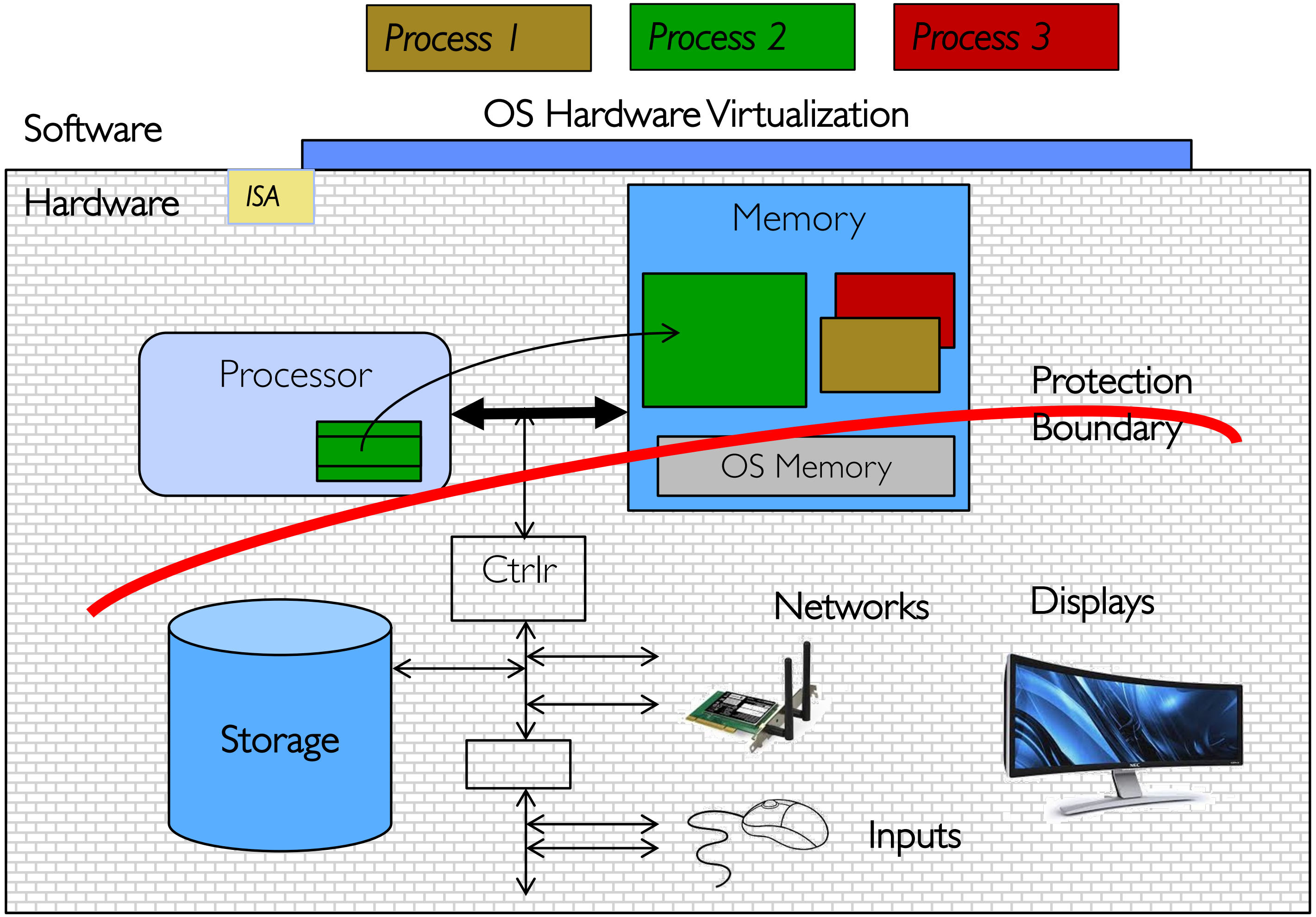
OS Basics: Protection



OS Basics: Protection



OS Basics: Protection



进程间

- OS *isolates* processes from each other

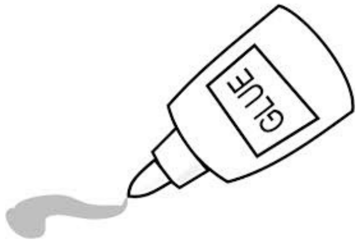
OS和进程间

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

What is an Operating System?

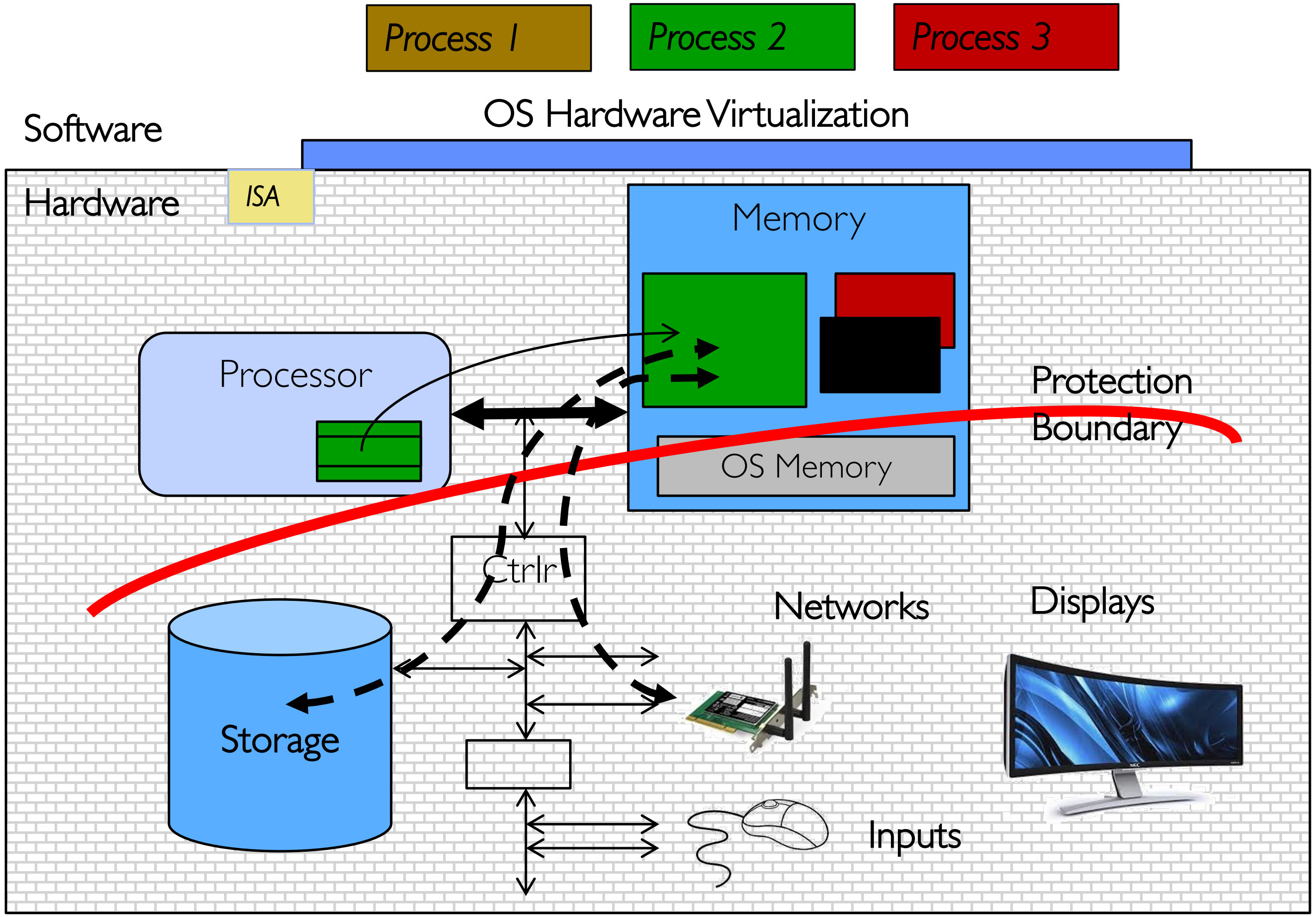
- Referee
  - ➢ Manage protection, isolation, and sharing of resources
    - » Resource allocation and communication

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



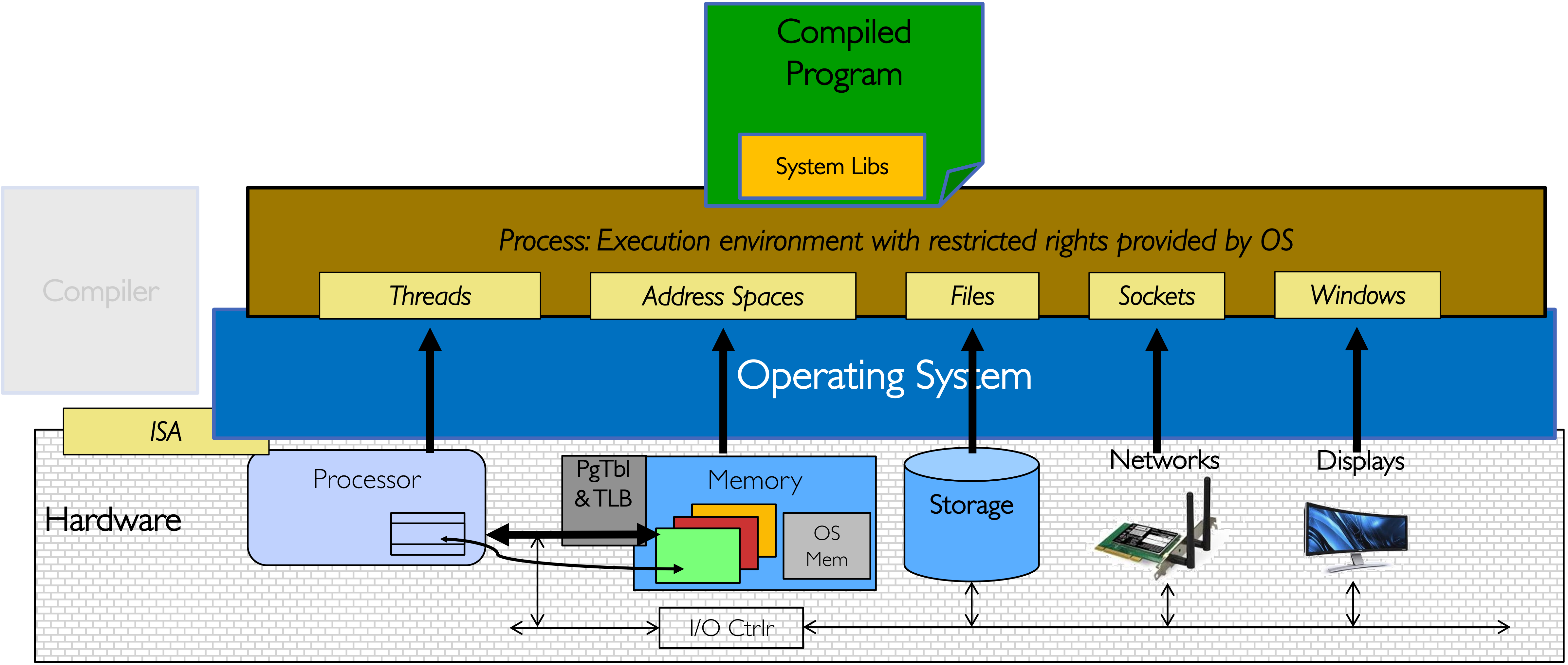
- Glue
  - ➢ Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

OS Basics: I/O

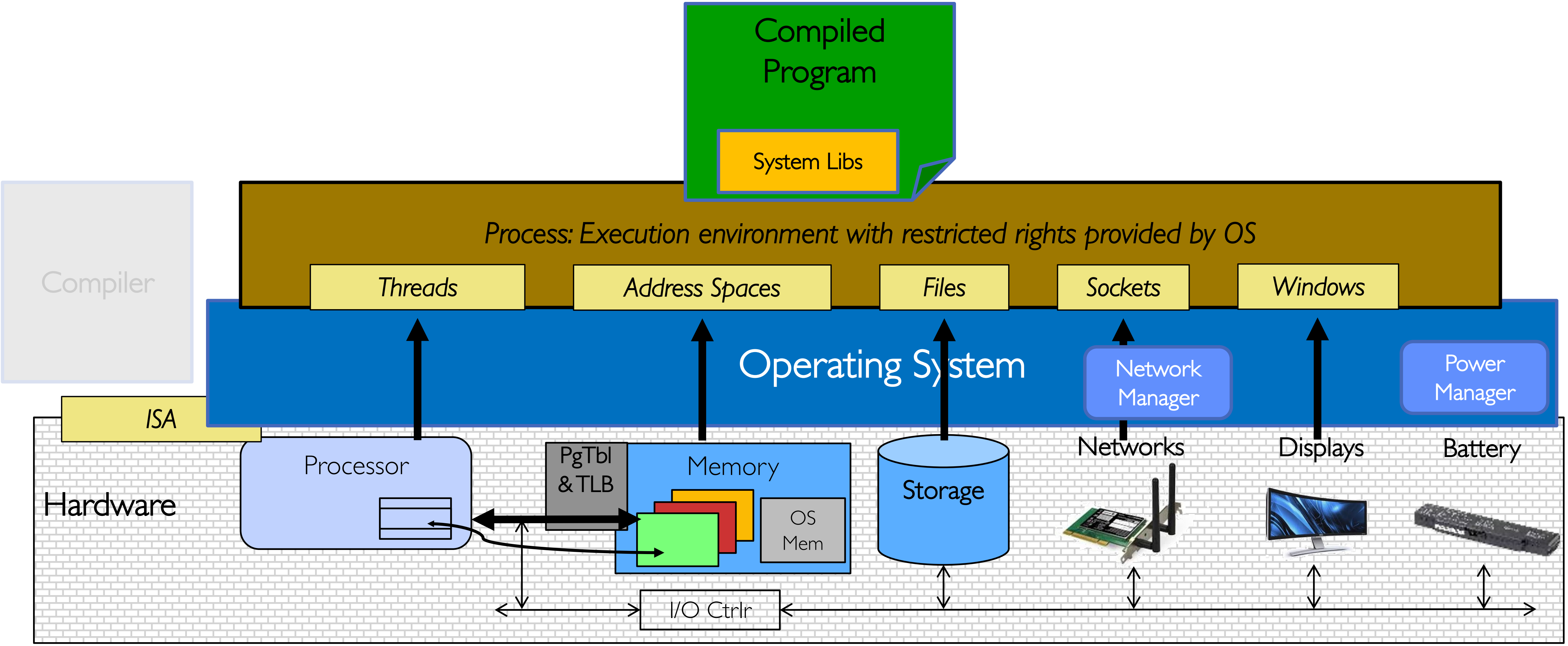


- OS provides common services in the form of I/O

OS Basics: Look and Feel



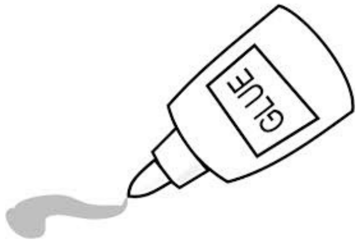
OS Basics: Background Management



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



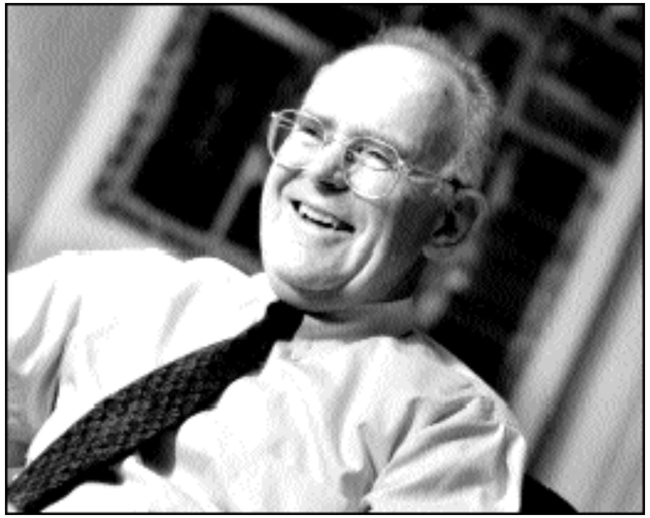
- Glue
  - ➢Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

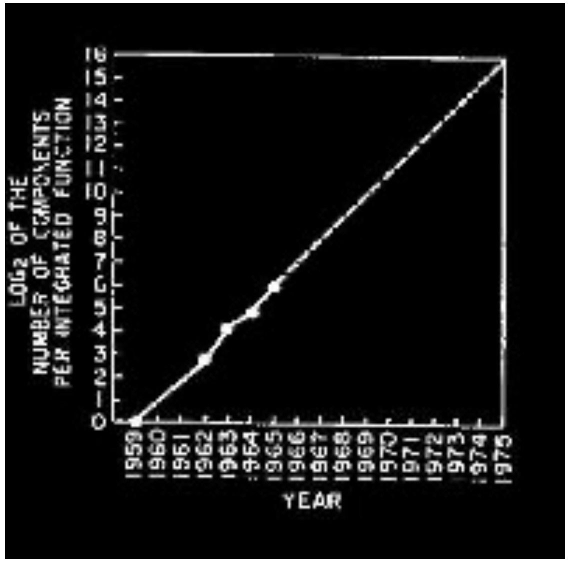
Why take Operating Systems?

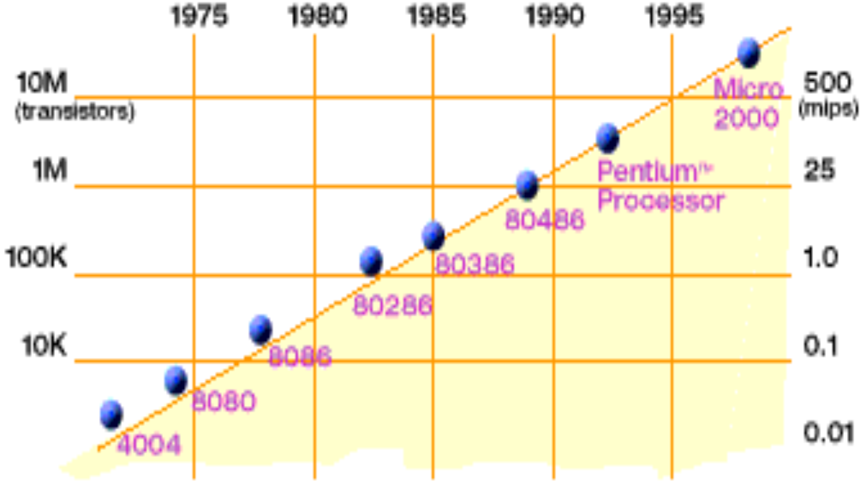
- Some of you will actually design and build operating systems or components of them.
  - Perhaps more now than ever
- Many of you will create systems that utilize the core concepts in operating systems.
  - Whether you build software or hardware
  - The concepts and design patterns appear at many levels
- 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"

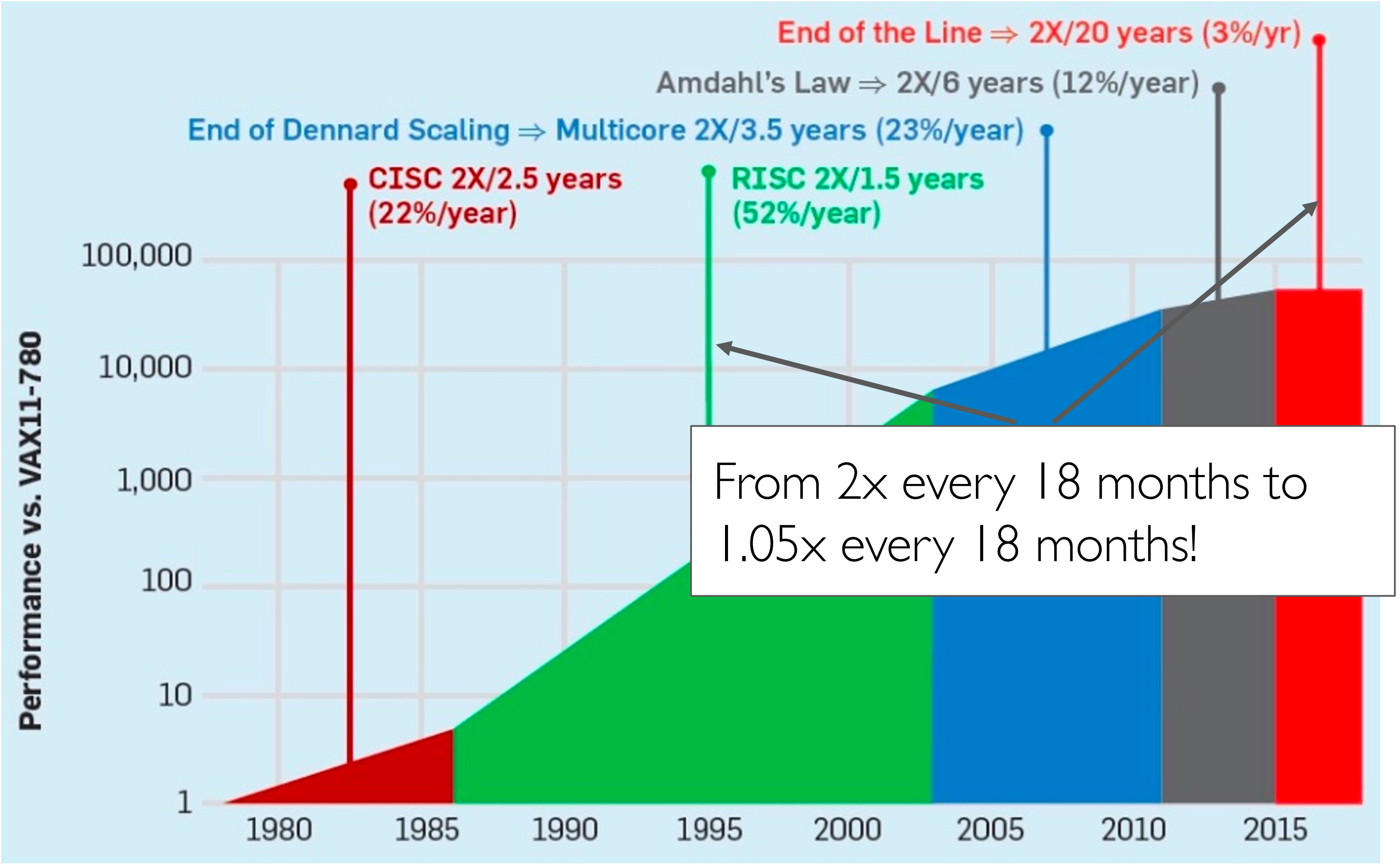
Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 2 years

- Microprocessors have become smaller, denser, and more powerful

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

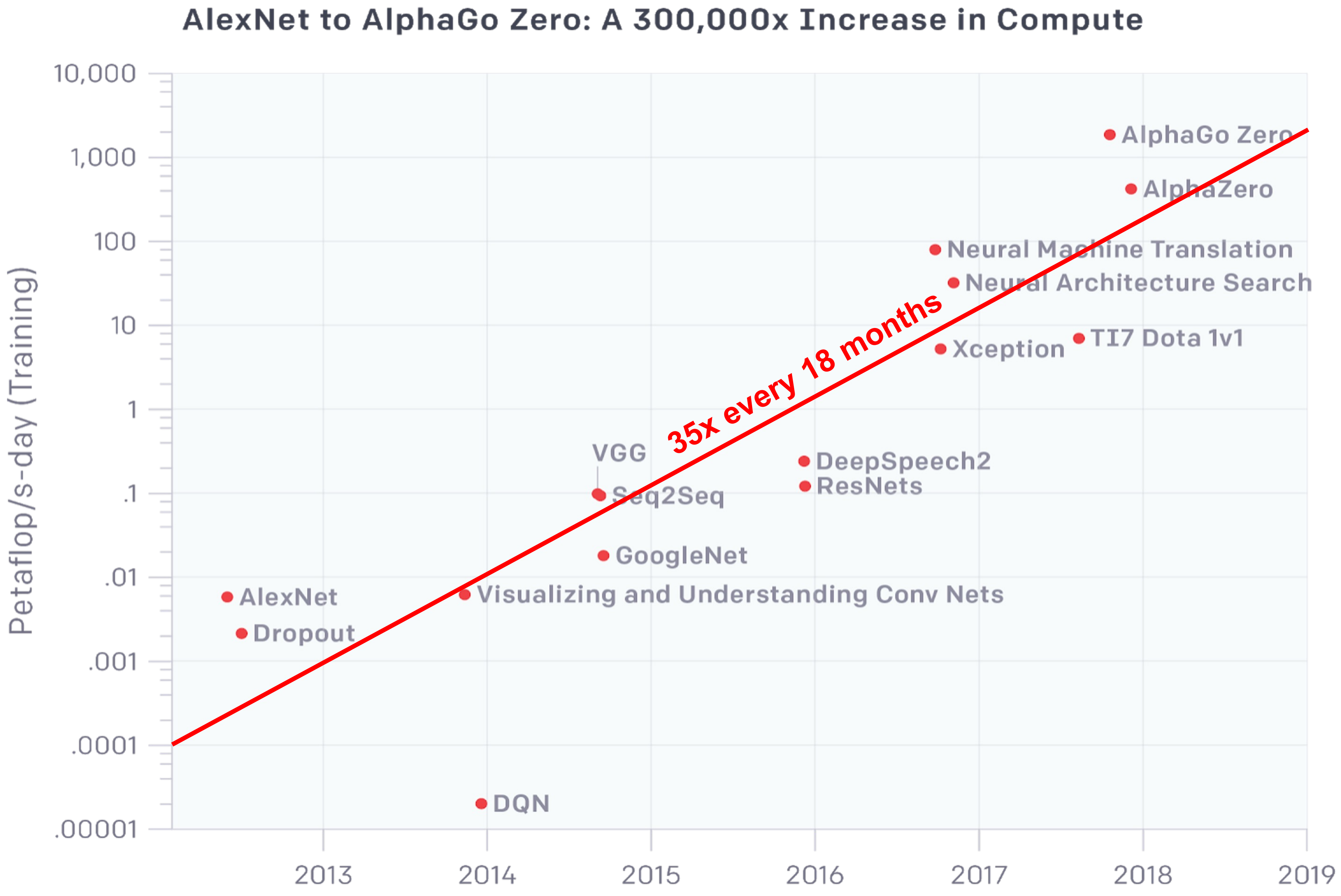
- Faster growth because transistors are closer to each other so electrical signals travel faster

The end of the Moore's Law



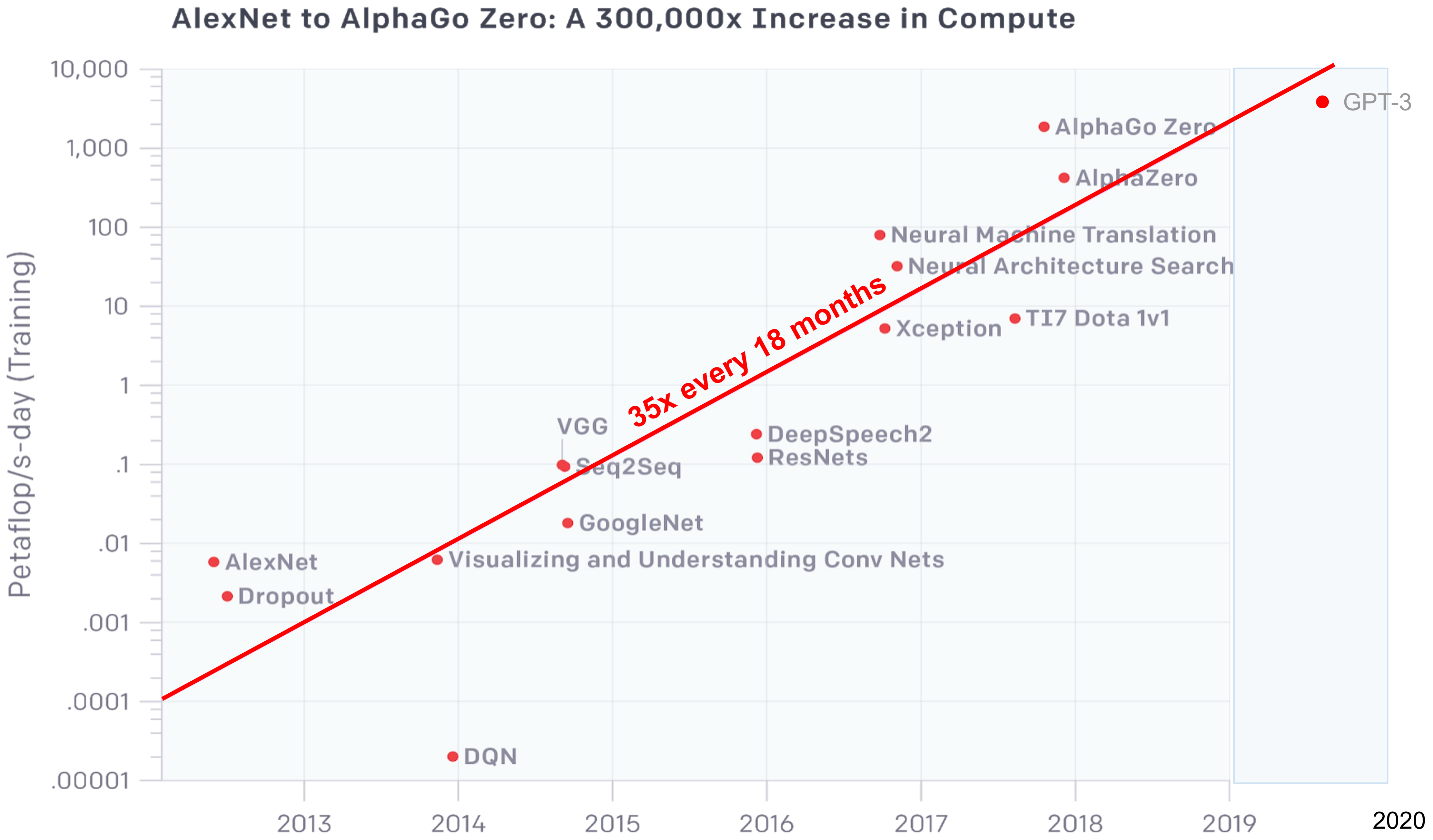
John L. Hennessy and David A. Patterson.A New Golden Age for Computer Architecture. Communications of the ACM 2019.

AI Compute demands: 2012 - 2019



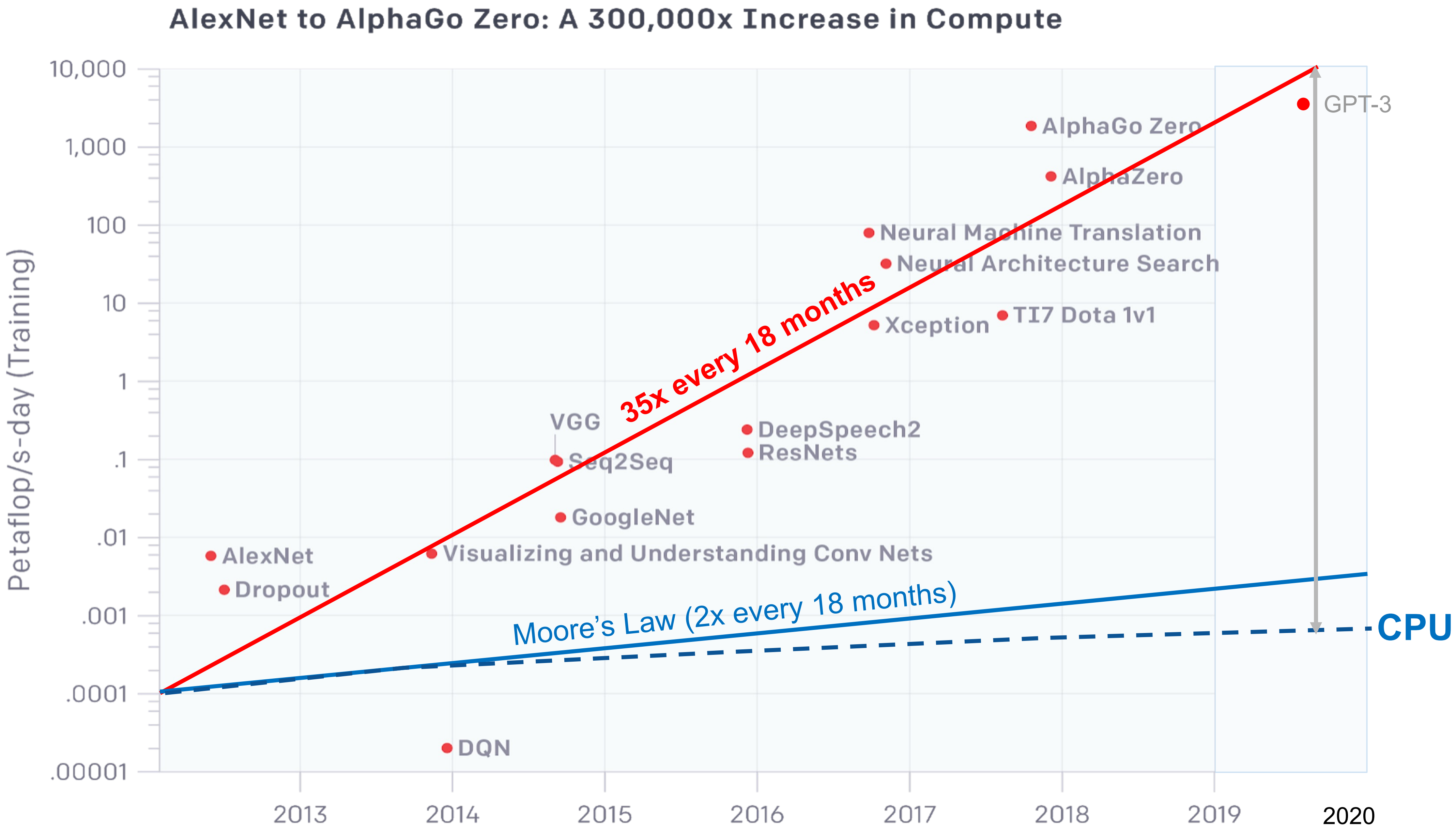
(https://openai.com/blog/ai-and-compute//)

AI Compute demands: 2012 – **2020**



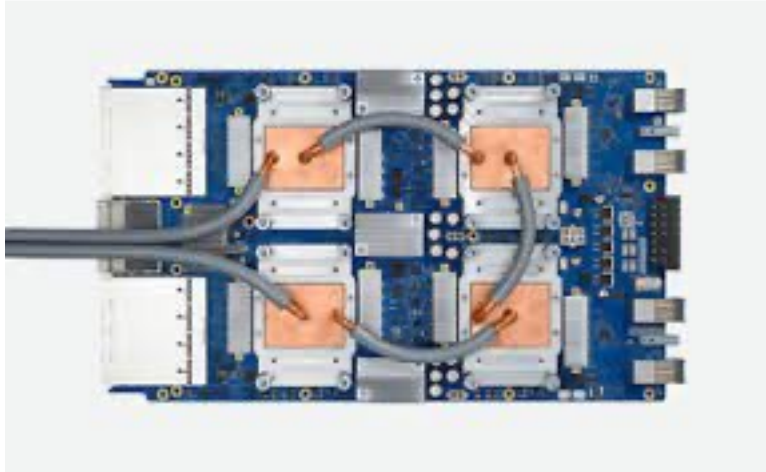
(https://openai.com/blog/ai-and-compute//)

Growing gap between demand and supply



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

Specialized processors to the rescue?

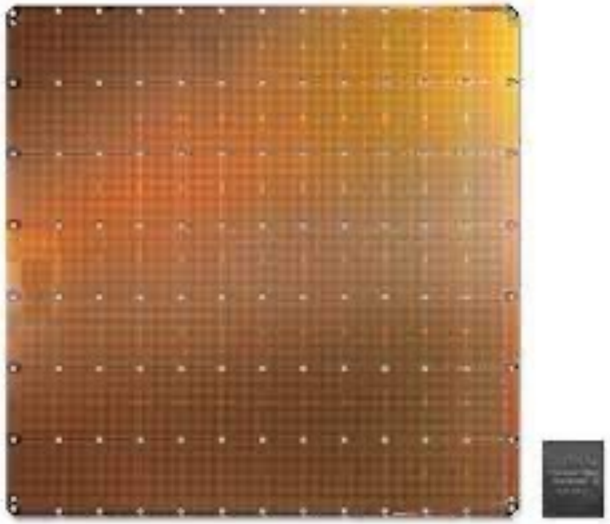


TPU v3

Nvidia A100

AWS's Inferentia

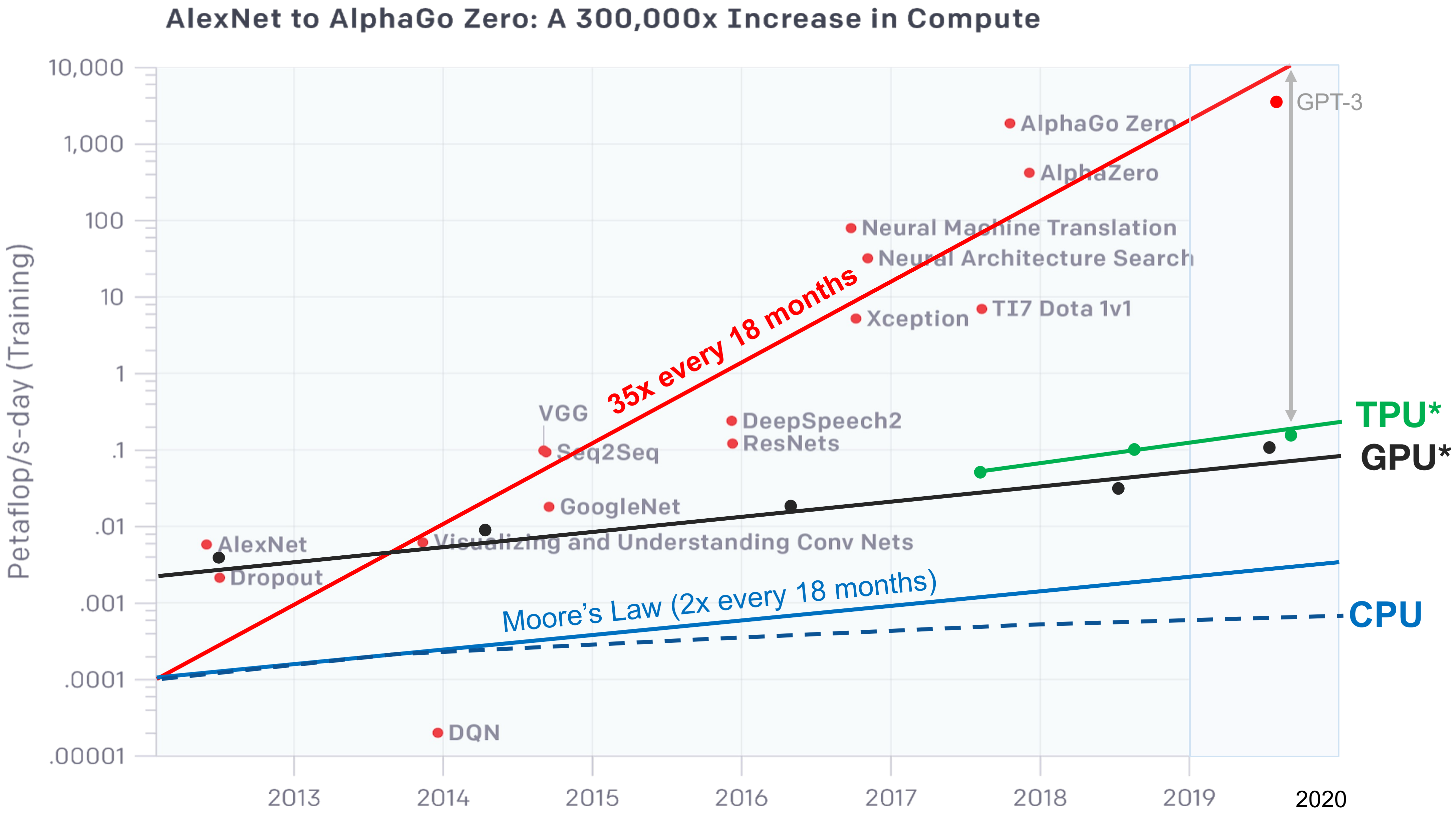
AMD Radeon



Cerebras

Intel Alchemist

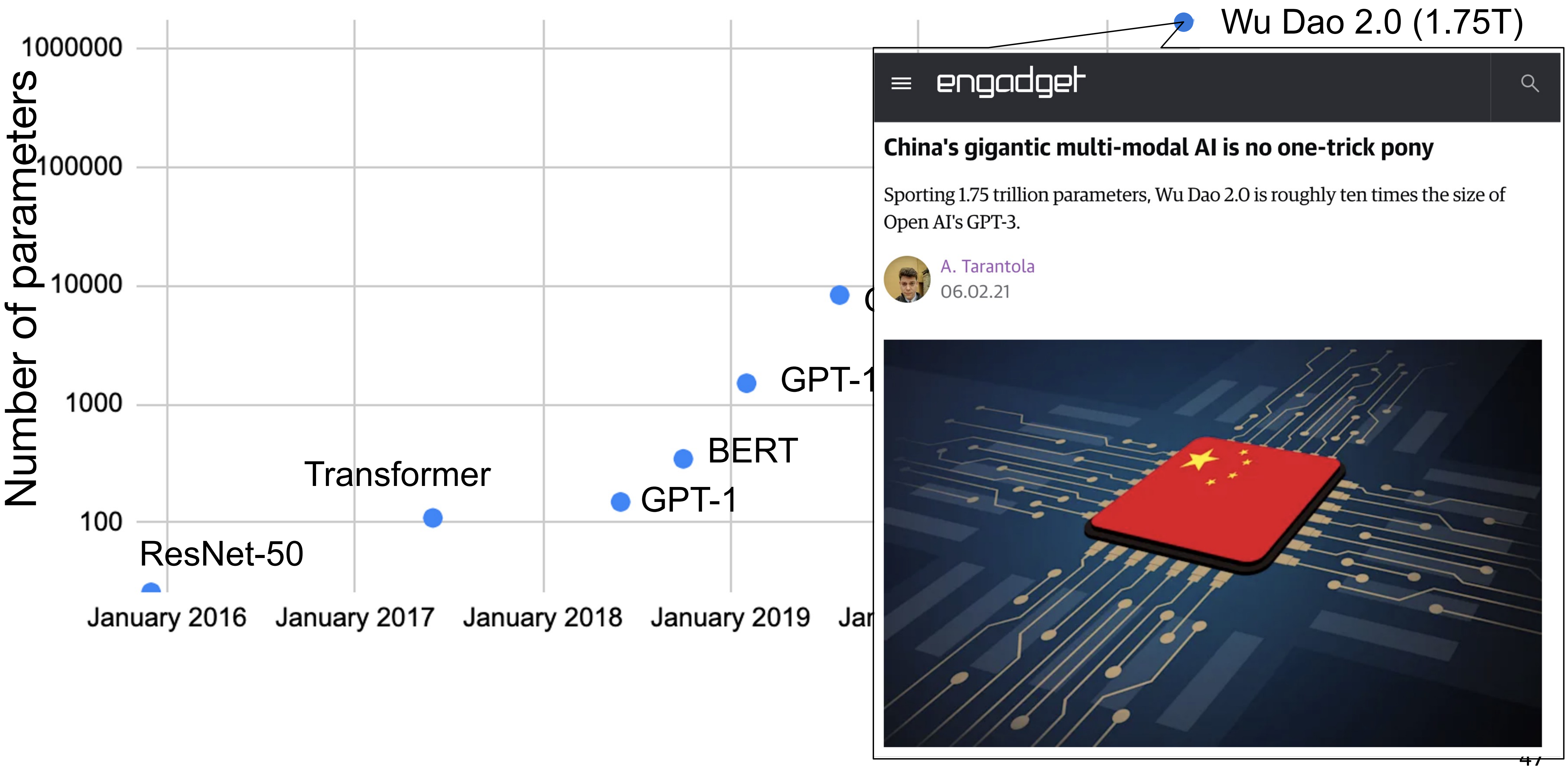
Specialized hardware note enough



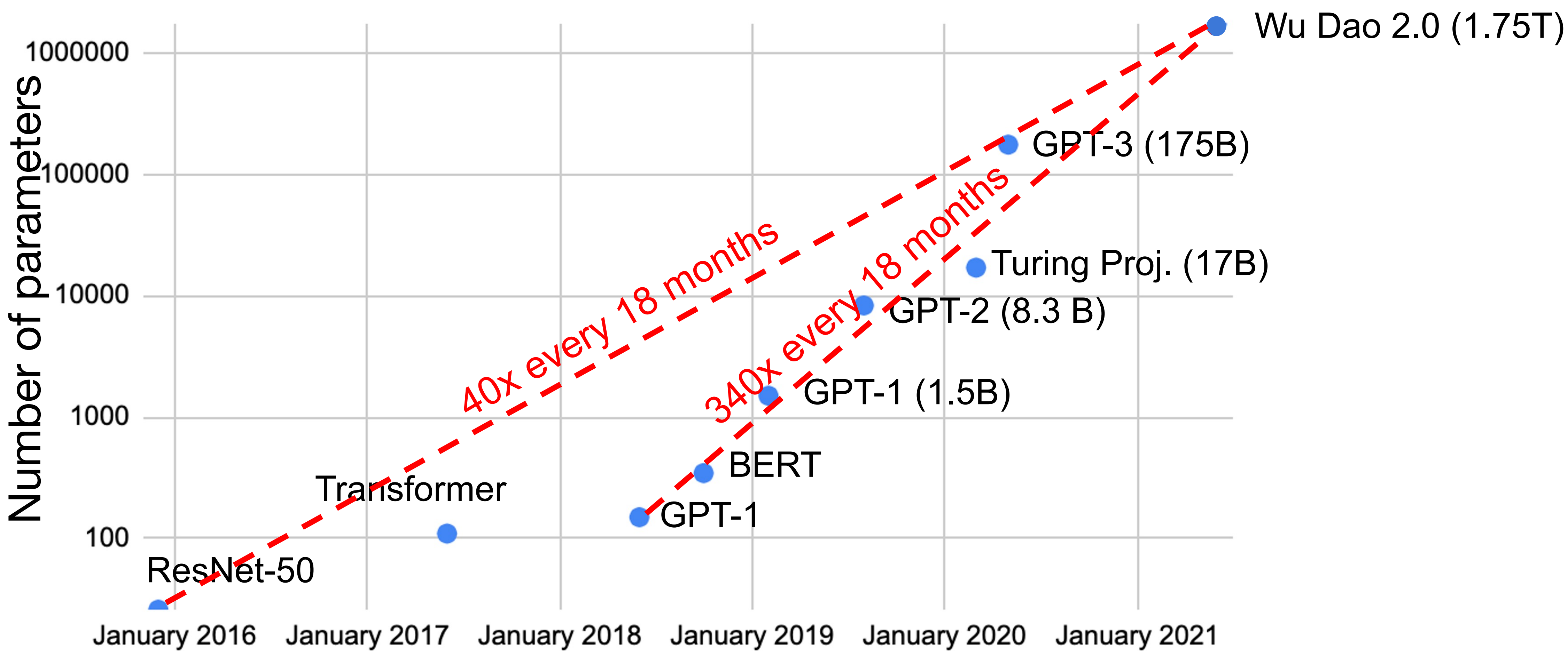
* assume 0.33 utilization

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

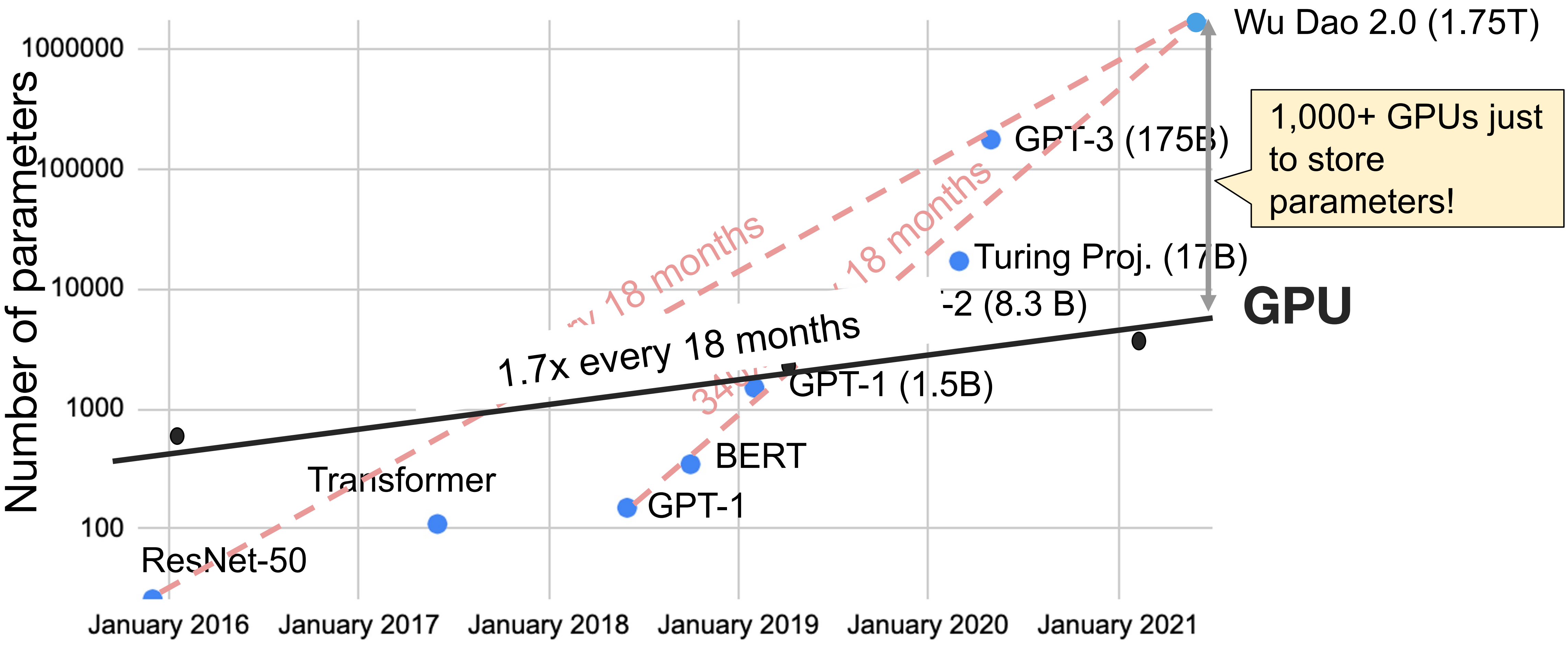
Memory demands growing as fast



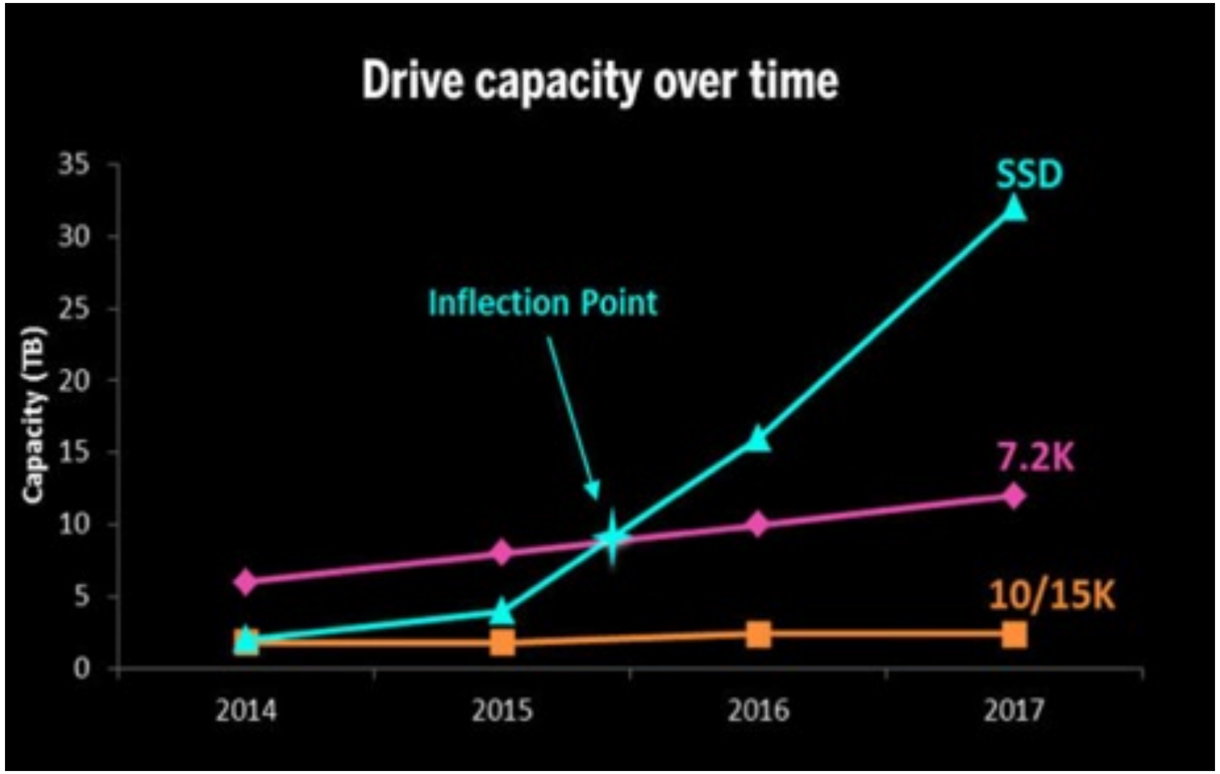
Memory demands growing as fast



Huge Gap between memory demands and single-chip memory



Storage Capacity is Still Growing!



Storage Capacity

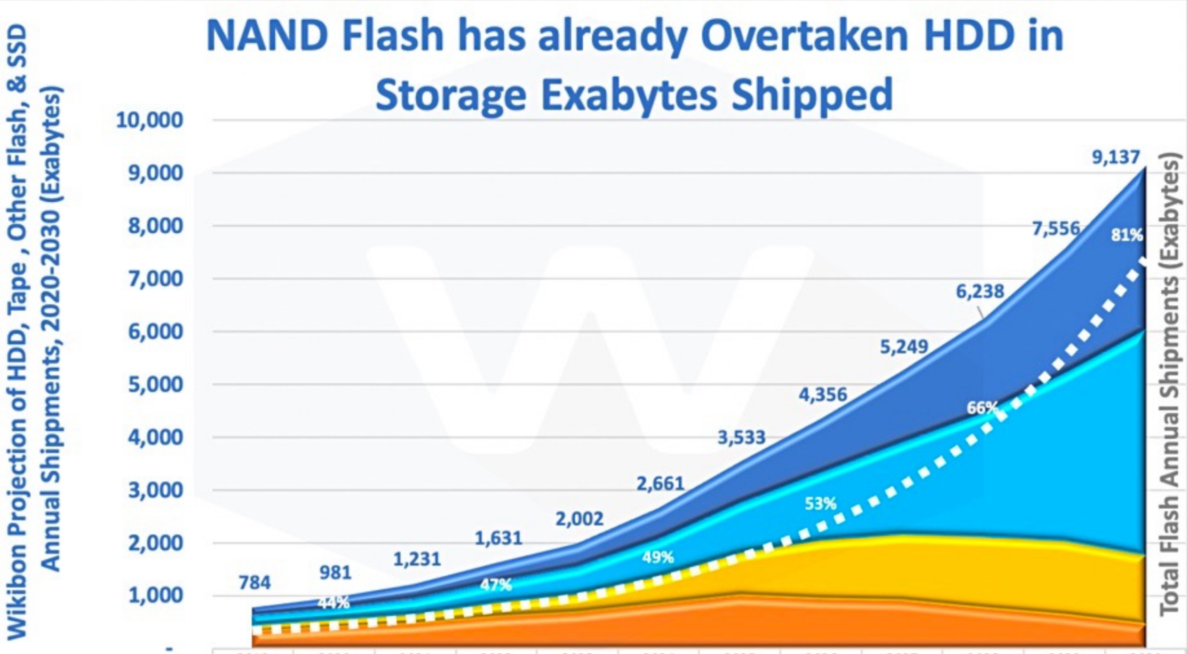
Largest SSD 3.5-inch drive:
100 TB @ $40K ($400/TB)

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

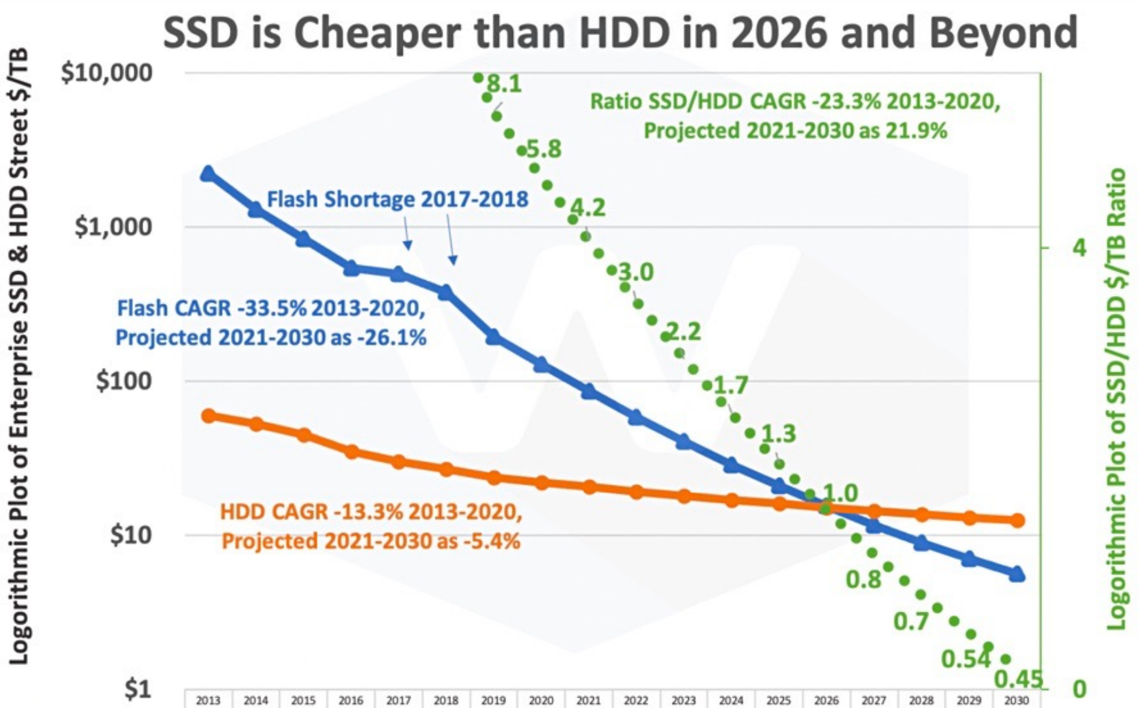


	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030
SSD Flash Exabytes	152	217	272	355	444	586	774	1021	1348	1779	2349	3100
Other Flash Exabytes	191	218	304	412	524	707	955	1289	1740	2350	3172	4282
Tape Exabytes	115	137	177	259	352	533	812	1,105	1,253	1,350	1,404	1,298
HDD Exabytes	326	410	478	605	682	834	992	941	907	759	631	456
Total Flash Exabytes	343	435	576	767	968	1,293	1,729	2,310	3,088	4,129	5,521	7,382
Total Exabytes	784	981	1,231	1,631	2,002	2,661	3,533	4,356	5,249	6,238	7,556	9,137

Source: © Wikibon, 2021

Figure 9 - Exabyte Storage Shipments Split by SSD, Other Flash, HDDs, and Tape

Source: © Wikibon, 2021.



	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030
SSD $/TB	$2,220	$1,293	$840	$538	$493	$378	$195	$128	$86	$58	$40	$29	$21	$15	$12	$9	$7	$6
HDD $/TB	$60	$53	$45	$35	$30	$26.8	$23.9	$22.2	$21	$19	$18	$17	$16	$15	$14	$14	$13	$13
Ratio SSD/HDD $/TB	37	25	19	15	16	14	8.1	5.8	4.2	3.0	2.2	1.7	1.3	1.0	0.8	0.7	0.54	0.45

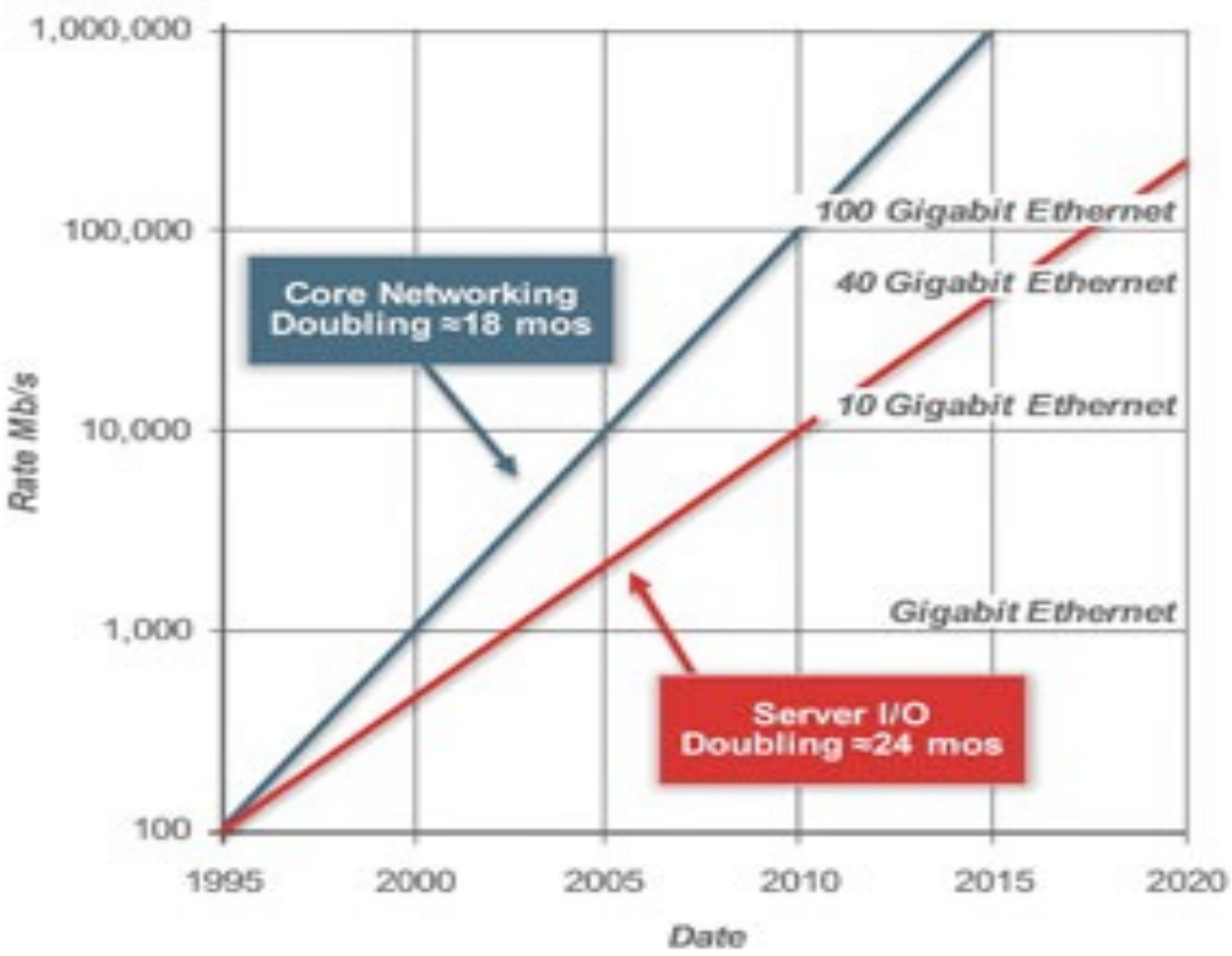
SSD $/TB HDD $/TB Ratio SSD/HDD $/TB

Source: © Wikibon, 2021. The 2021-2030 Enterprise Street Prices are Projected by Wikibon using Wright's Law

Figure 4 - SSD/HDD Pricing Ratio 2013 - 2030

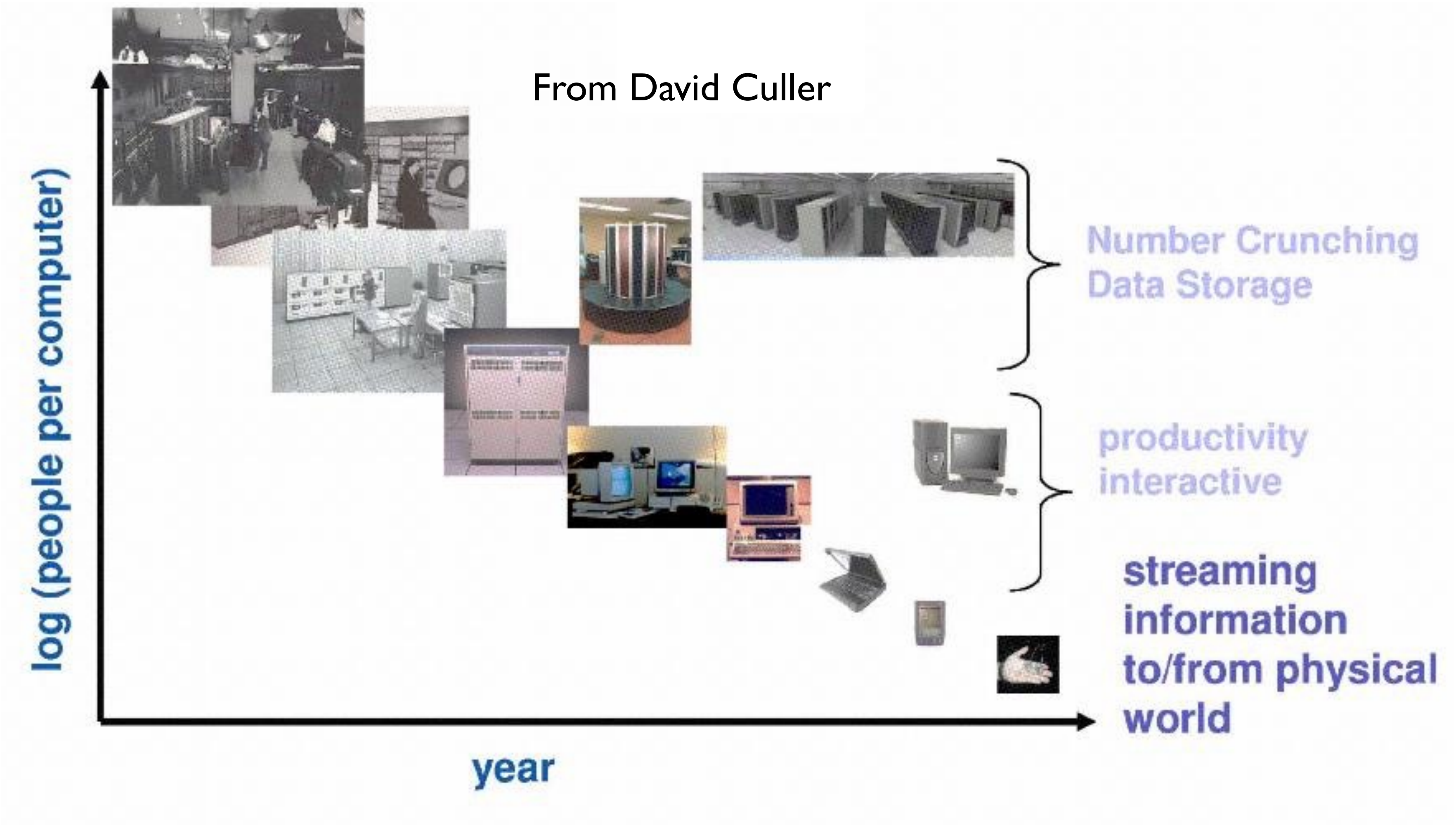
Source: © Wikibon, 2021.

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"

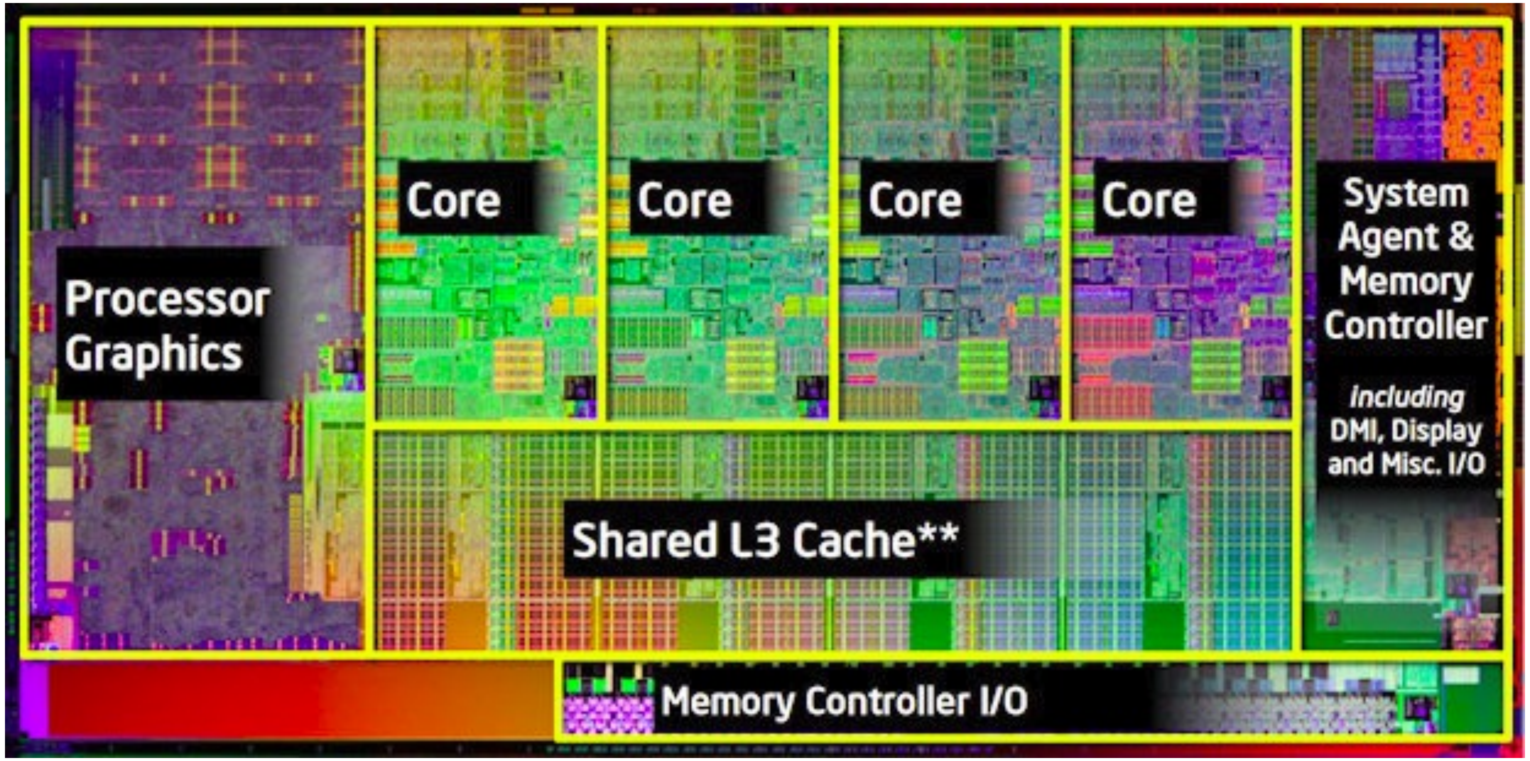
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk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

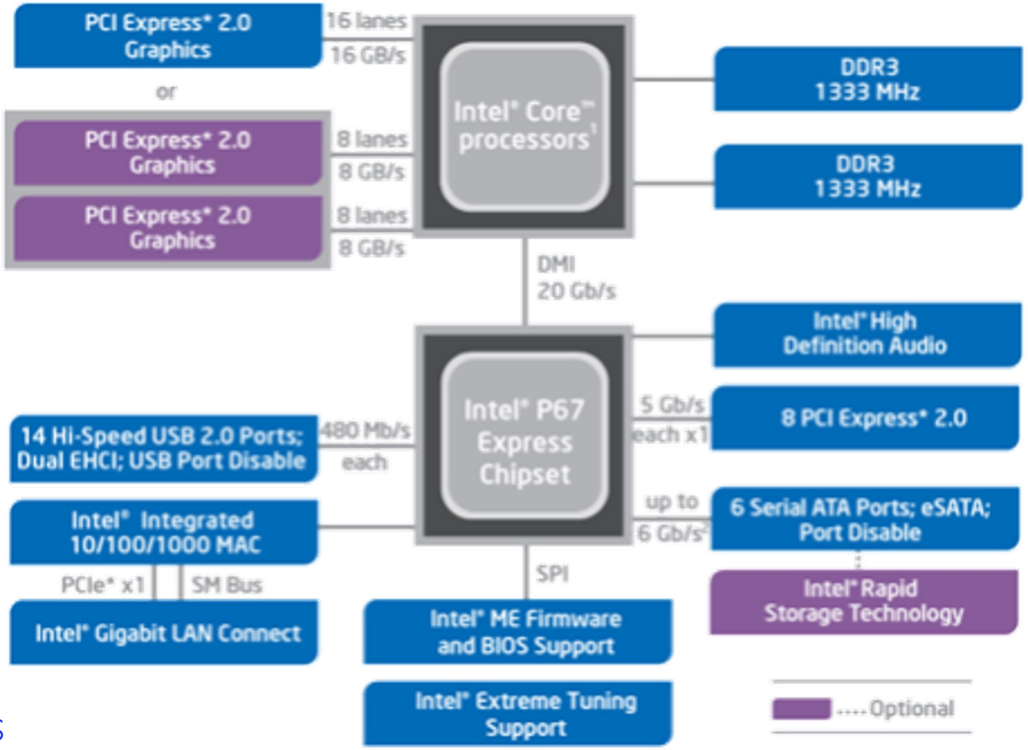
A Modern Processor: Intel Sandy Bridge

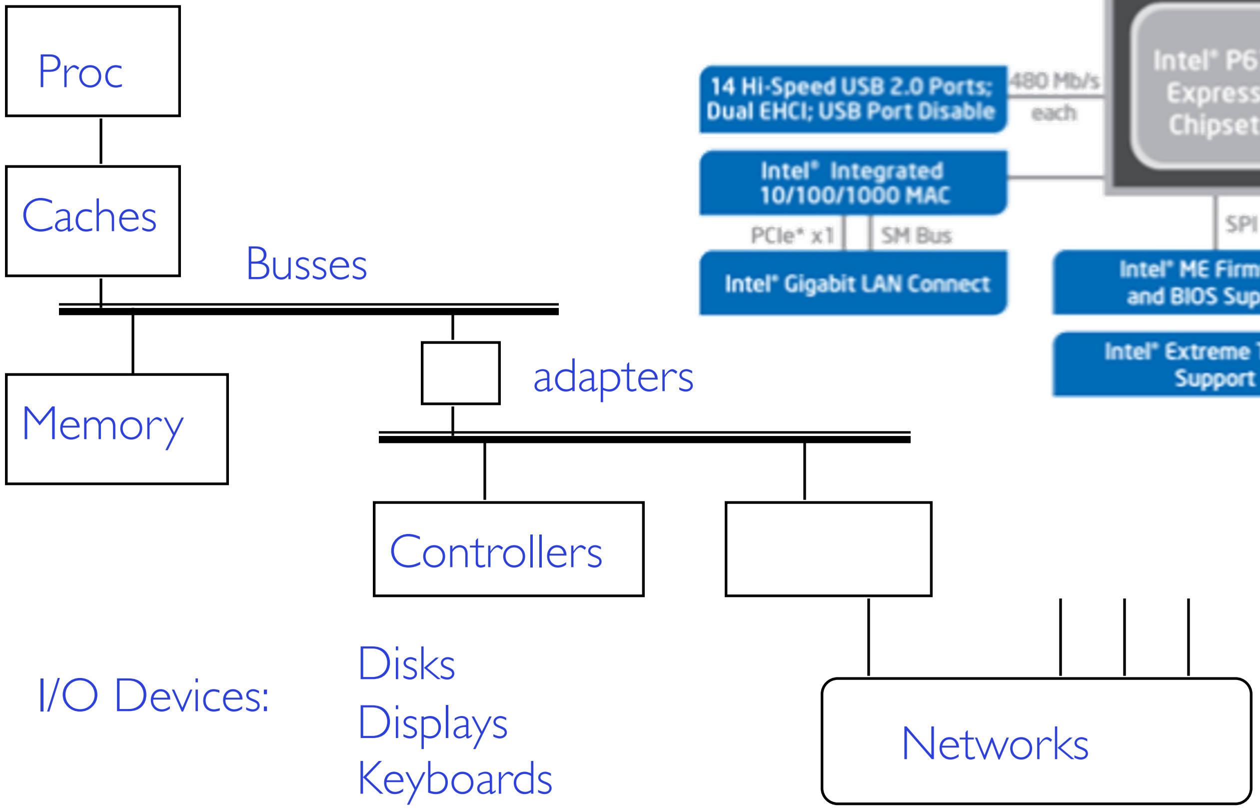


- Package: LGA 1155
  - 1155 pins
  - 95W design envelope
- Cache:
  - L1: 32K Inst, 32K Data (3 clock access)
  - L2: 256K (8 clock access)
  - Shared L3: 3MB – 20MB
- Transistor count:
  - 504 Million (2 cores, 3MB L3)
  - 2.27 Billion (8 cores, 20MB L3)
- Note that ring bus is on high metal layers – above the Shared L3 Cache

HW Functionality comes with great complexity!

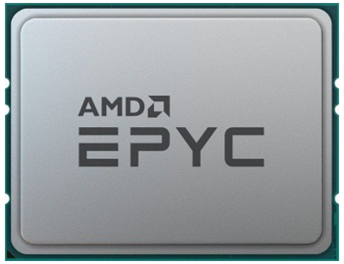
Intel Sandy Bridge I/O Configuration



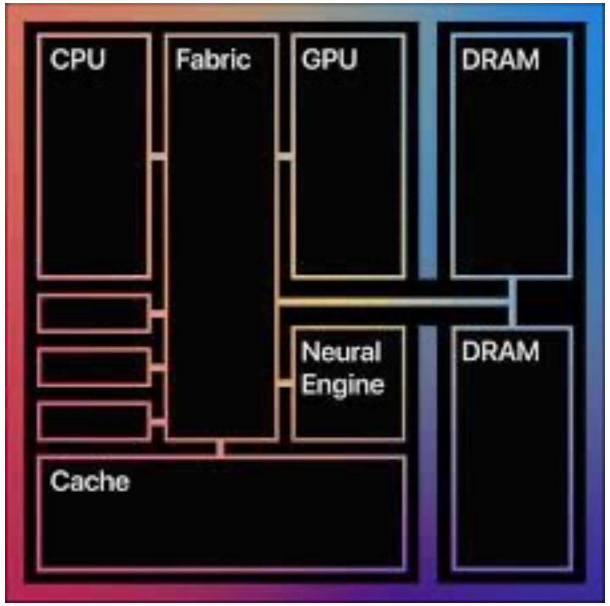


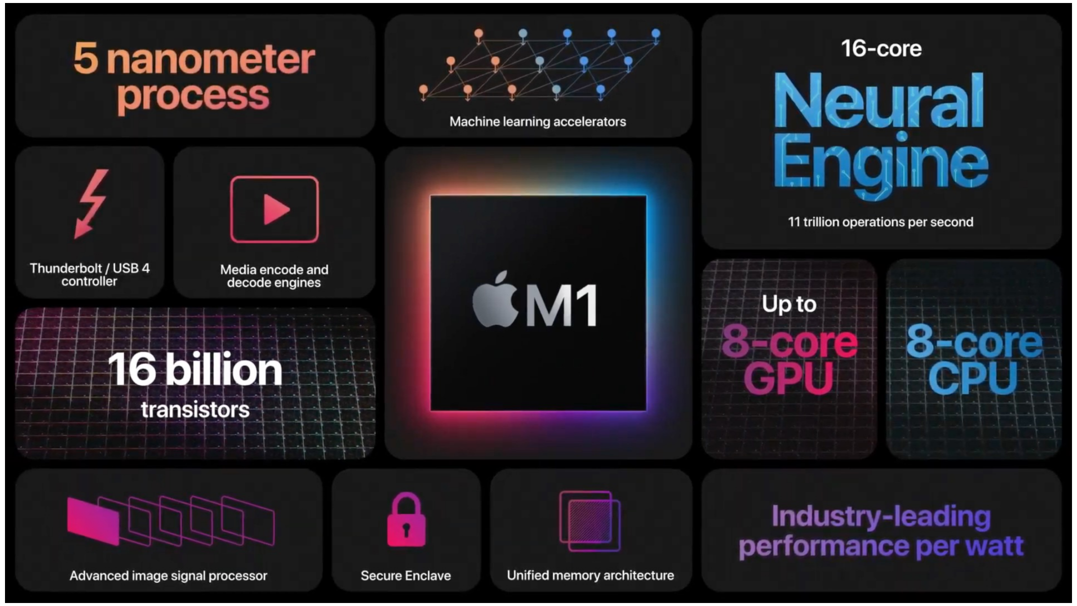
Not only specialized processors…

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

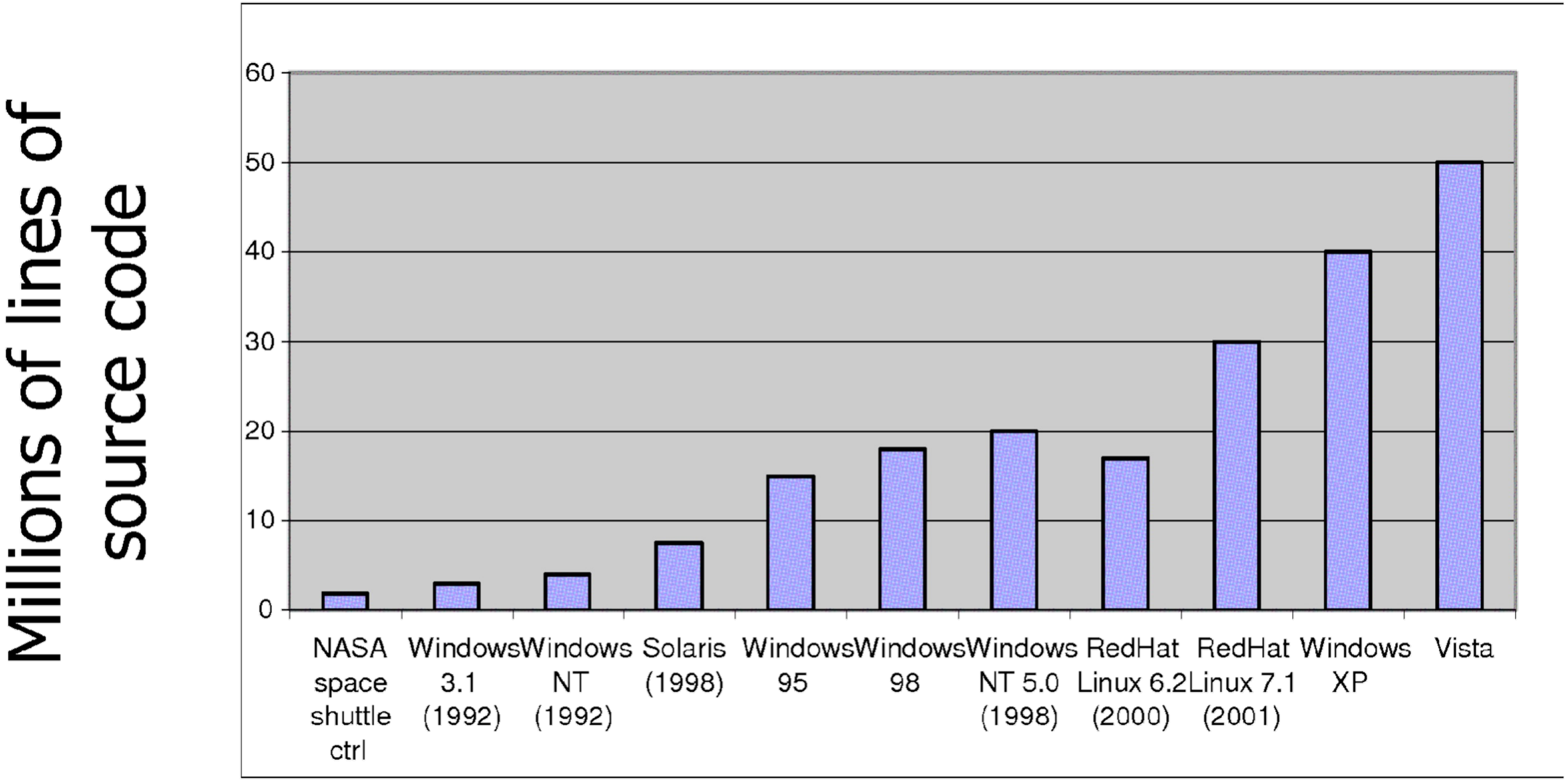


- System On a Chip (SOC): accelerating trend





Increasing Software Complexity



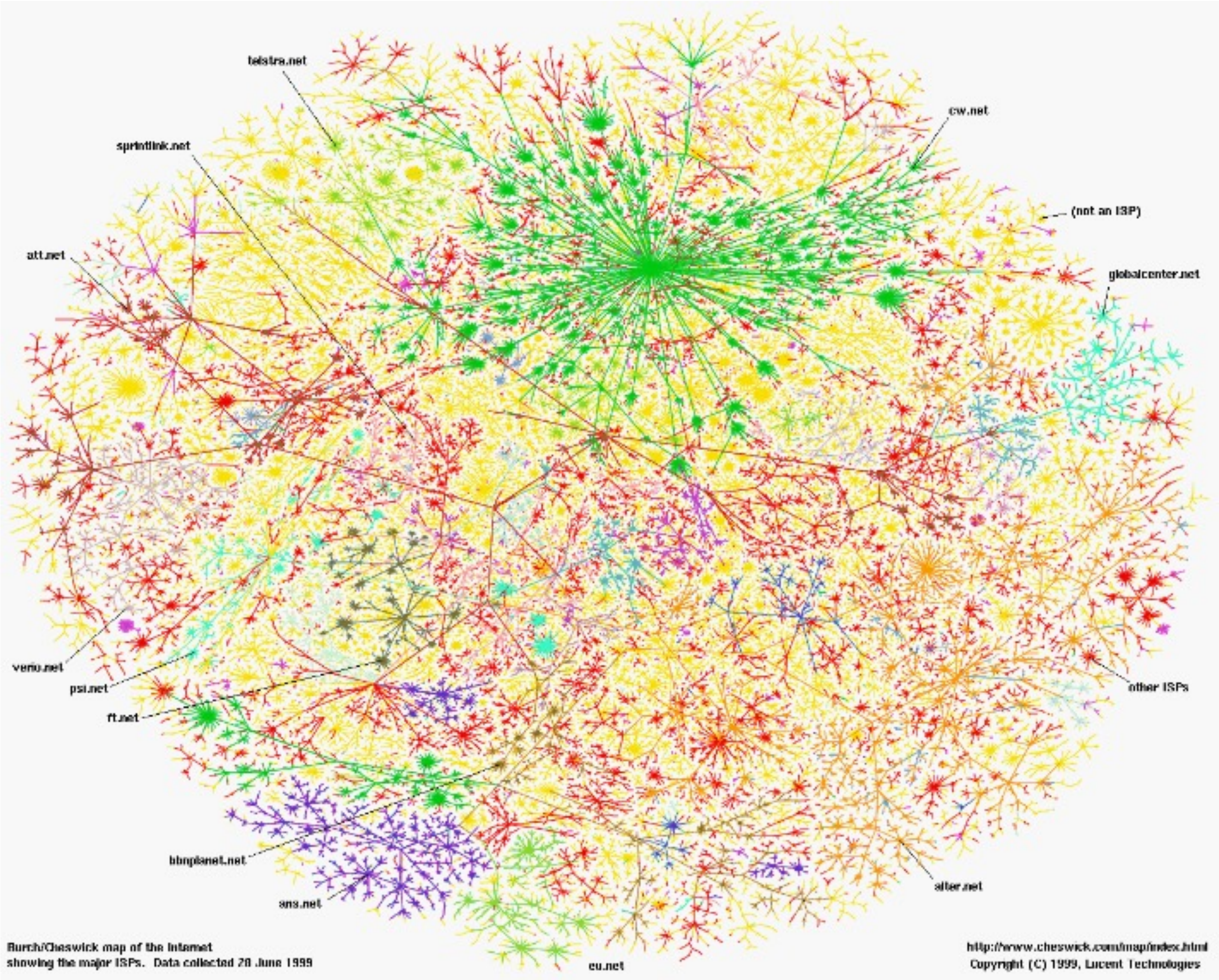
From MIT's 6.033 course

Everything going distributed

- Need to scale
  - ➢ Machine learning workloads
  - ➢ Big Data analytics
  - ➢ Scientific computing
  - ➢ …

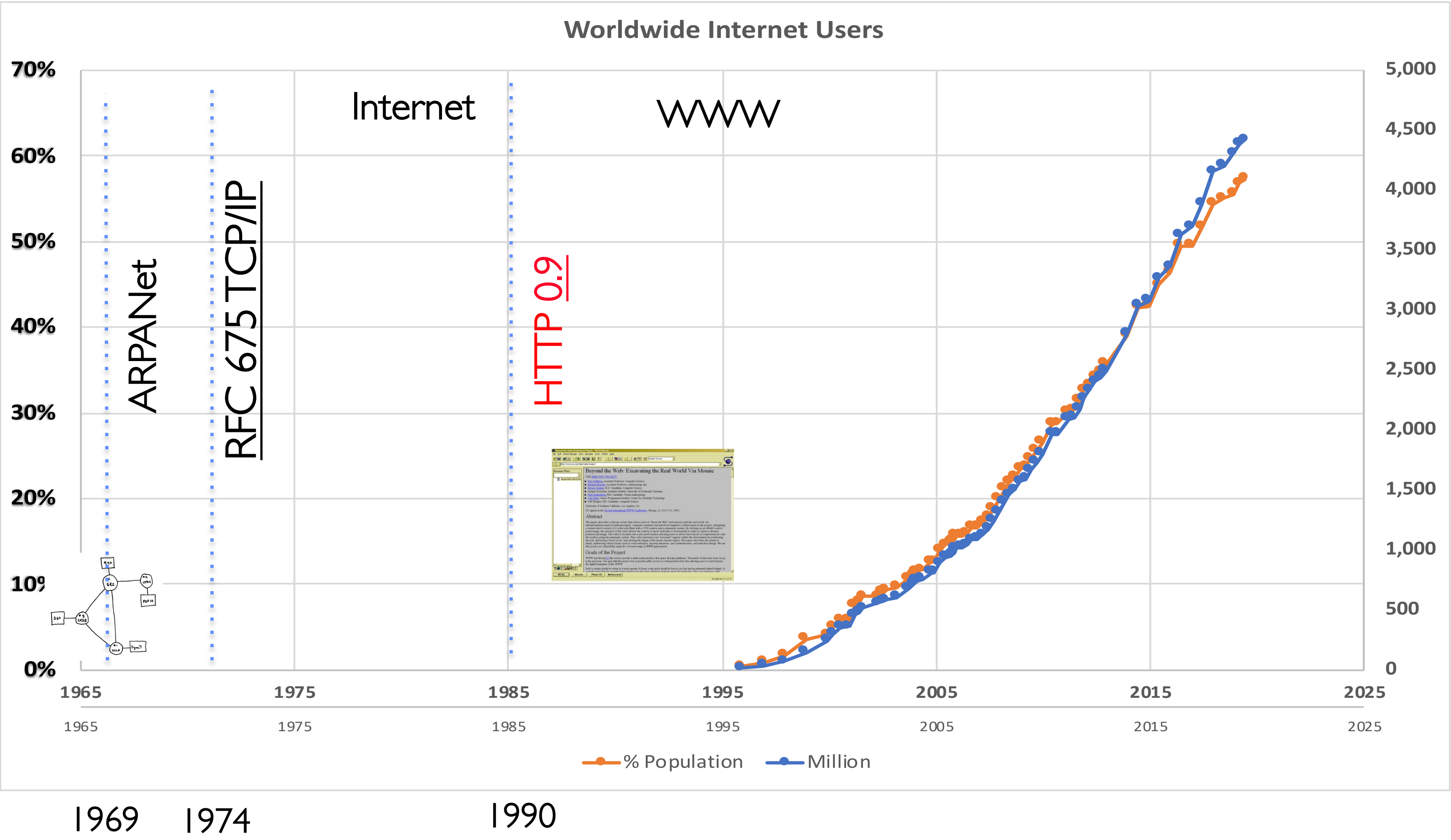
- Everything is connected!

Greatest Artifact of Human Civilization…



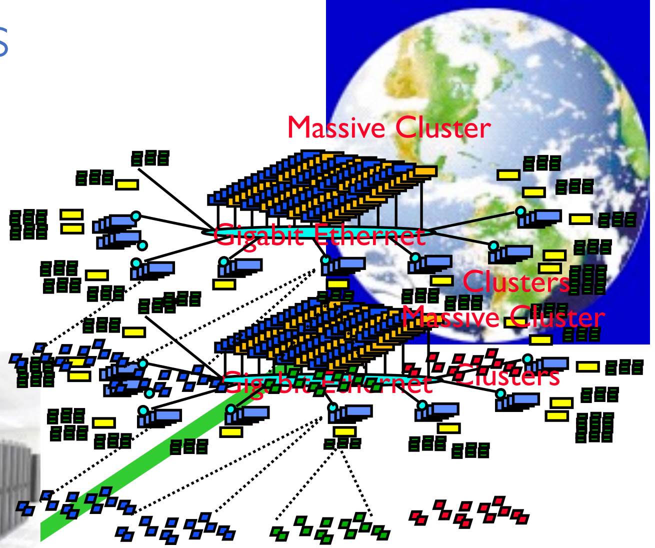
Map of the Internet

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



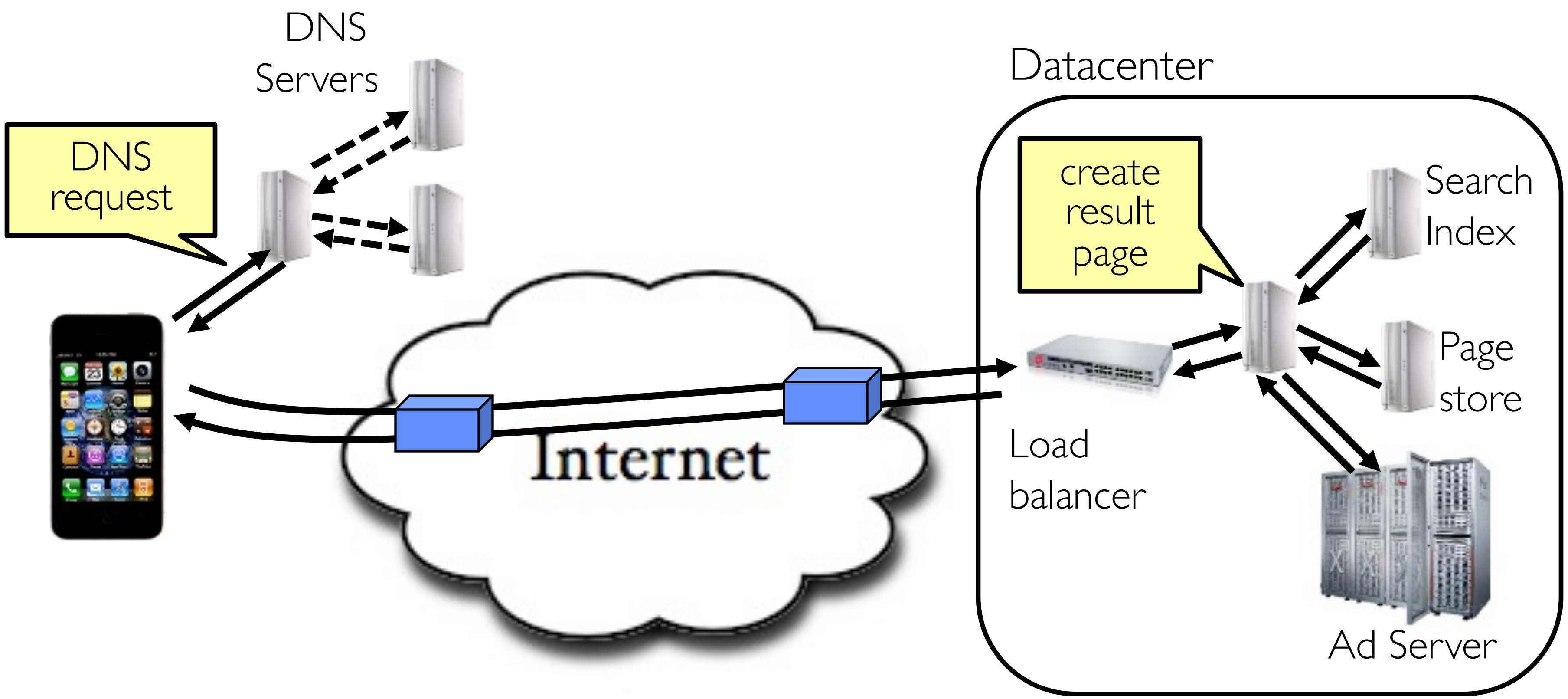
Scalable, Reliable,
Secure Services

Internet
Connectivity

Databases
Information Collection
Remote Storage
Online Games
Commerce
…

MEMS for
Sensor Nets

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 recognotion
  - 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