

Operating Systems (Honor Track)

What is an Operating System?

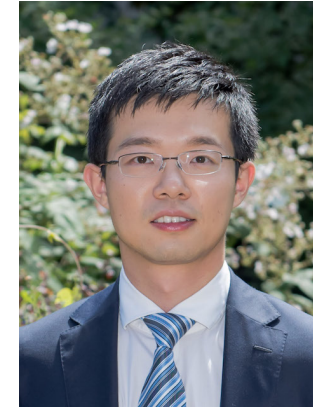
Xin Jin

Spring 2024

Course Staff

- Instructor: Xin Jin

- Research interests: Computer systems and networking
 - » Serverless computing
 - » Training and serving large language models
 - » Network testing and verification
 - » Disaggregated storage with RDMA and DPUs



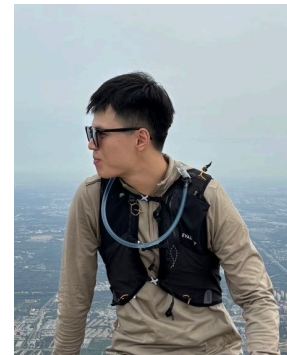
- Teaching assistants



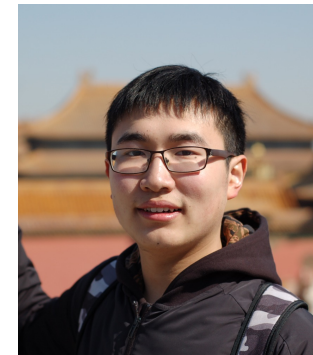
Chao Jin



Yingbo Sun



Zili Zhang

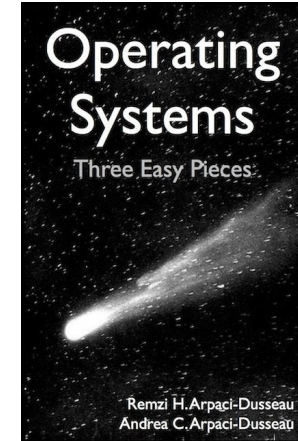


Yinmin Zhong

Infrastructure & Recommended Readings

- Infrastructure

- Website: <https://pku-os.github.io/sp24>
- Piazza: <https://piazza.com/pku.edu.cn/spring2024/04834490>
- Course feedback: <https://www.wjx.cn/vm/PZaRCka.aspx>
- Office hour: Friday 4-5pm, Yanyuan Building 811, or by appointment

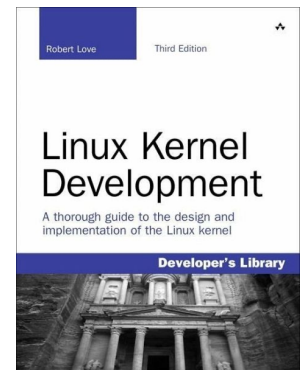
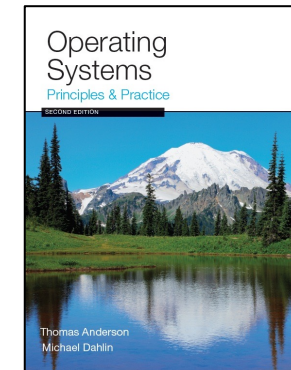


- Recommended textbook: Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau

- 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, LLM training and serving, 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
- Pintos from Stanford (used by OS courses at Stanford, Berkeley, CMU, etc.)
- **New this year: a new instructional OS based on Rust and RISC-V 😊**

- 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
- Reliability & Security
 - Fault tolerance, protection, security
- Cloud Infrastructure

Grading

- Participation: 5%
- Programming assignments (5 labs): 30%
 - 5 labs: 4% + 6.5% + 6.5% + 6.5% + 6.5%
 - 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: 25%
- Final exam: 40%

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

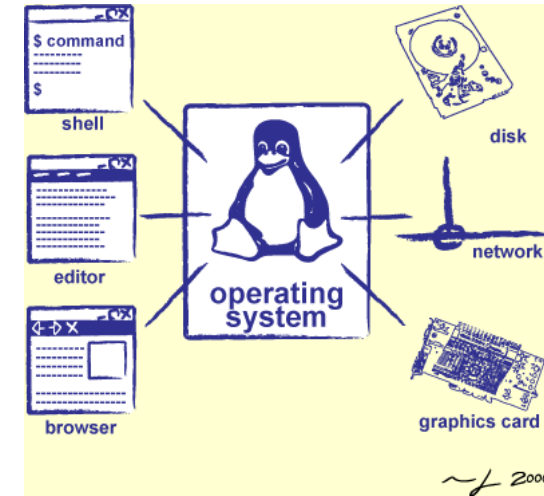
- Feedback survey: <https://www.wjx.cn/vm/PZaRCka.aspx>
 - Feel free to send us your feedback about the course.
 - It is available throughout the semester.
 - It is anonymous. A private channel to talk to me.
 - Your suggestions, comments, concerns and questions are **very valuable**.
 - **We will summarize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



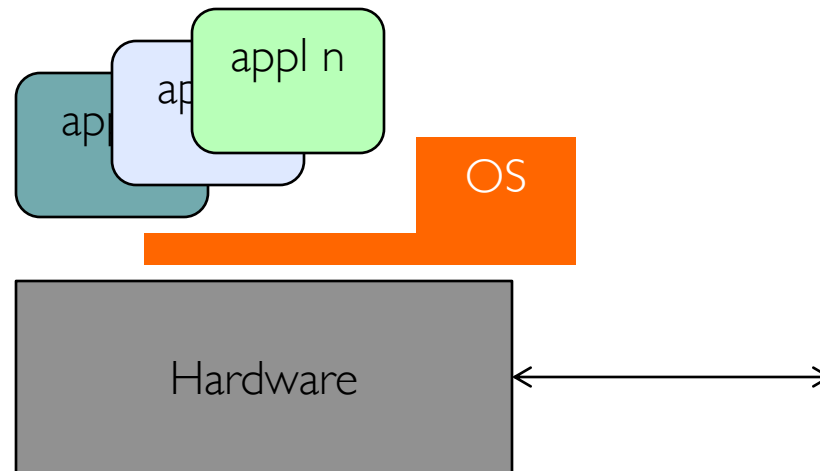
Interactive is important!

Ask Questions!

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

What is an operating system?

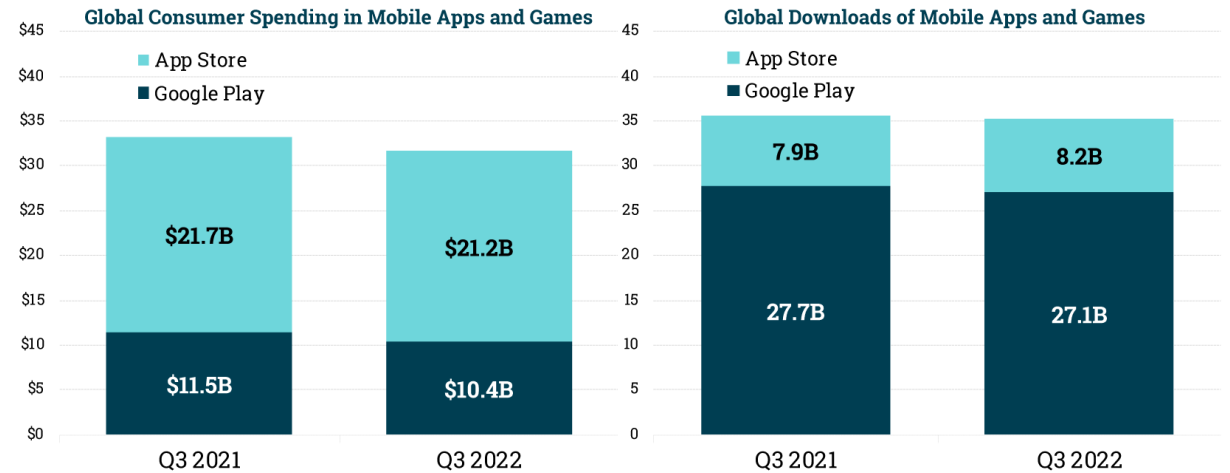
- Special layer of software that provides application software access to hardware resources
 - Convenient abstraction of complex hardware devices
 - Protected access to shared resources
 - Security and authentication
 - Communication amongst logical entities



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
 - ...

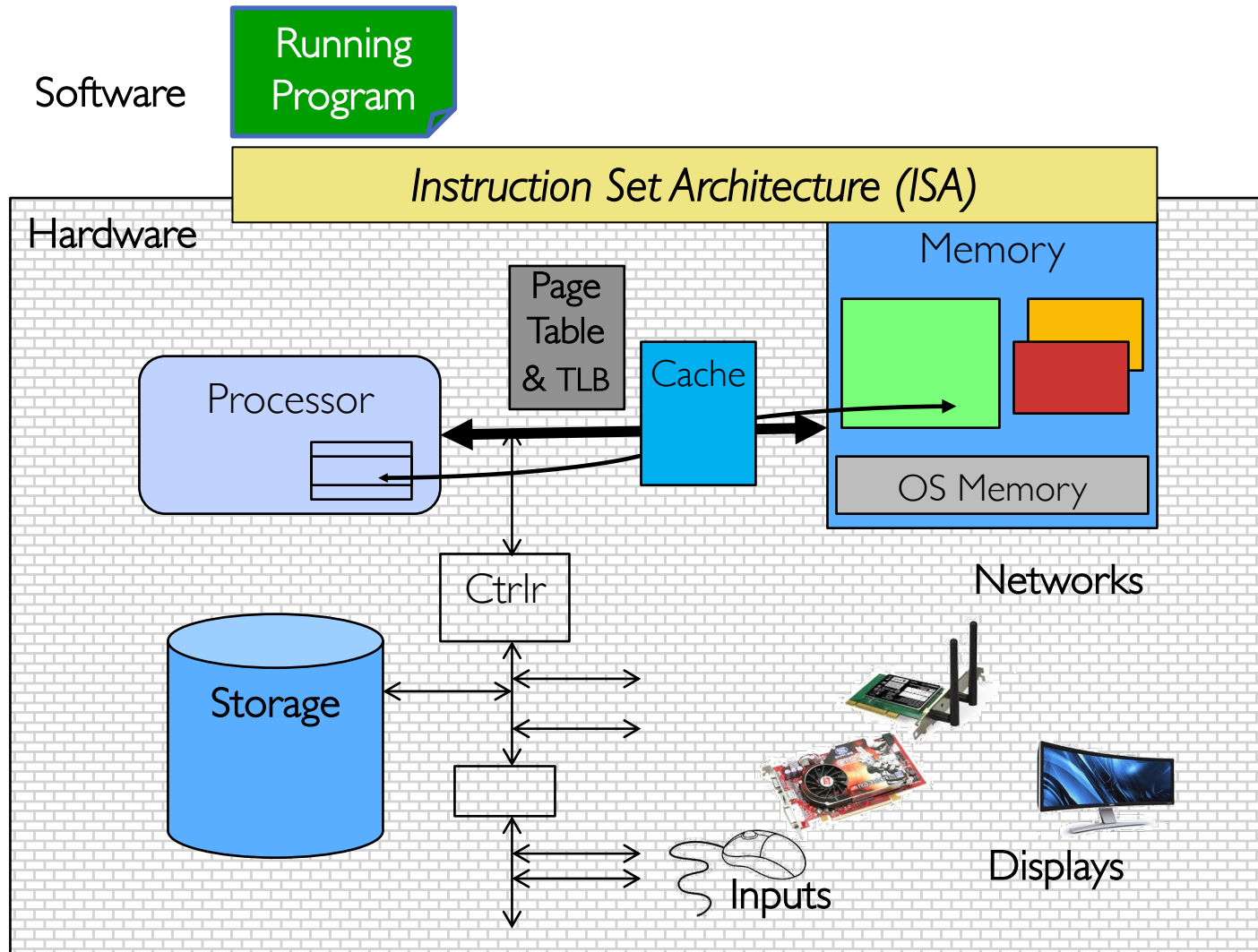
Global Mobile App Spending and Downloads in Q3 2022



Published on MarketingCharts.com in October 2022 | Data Source: Sensor Tower Store Intelligence

Estimated data for the periods between July 1 and September 30 2022 and 2021

Hardware/Software Interface



What you learned in Introduction to Computer Systems (ICS)

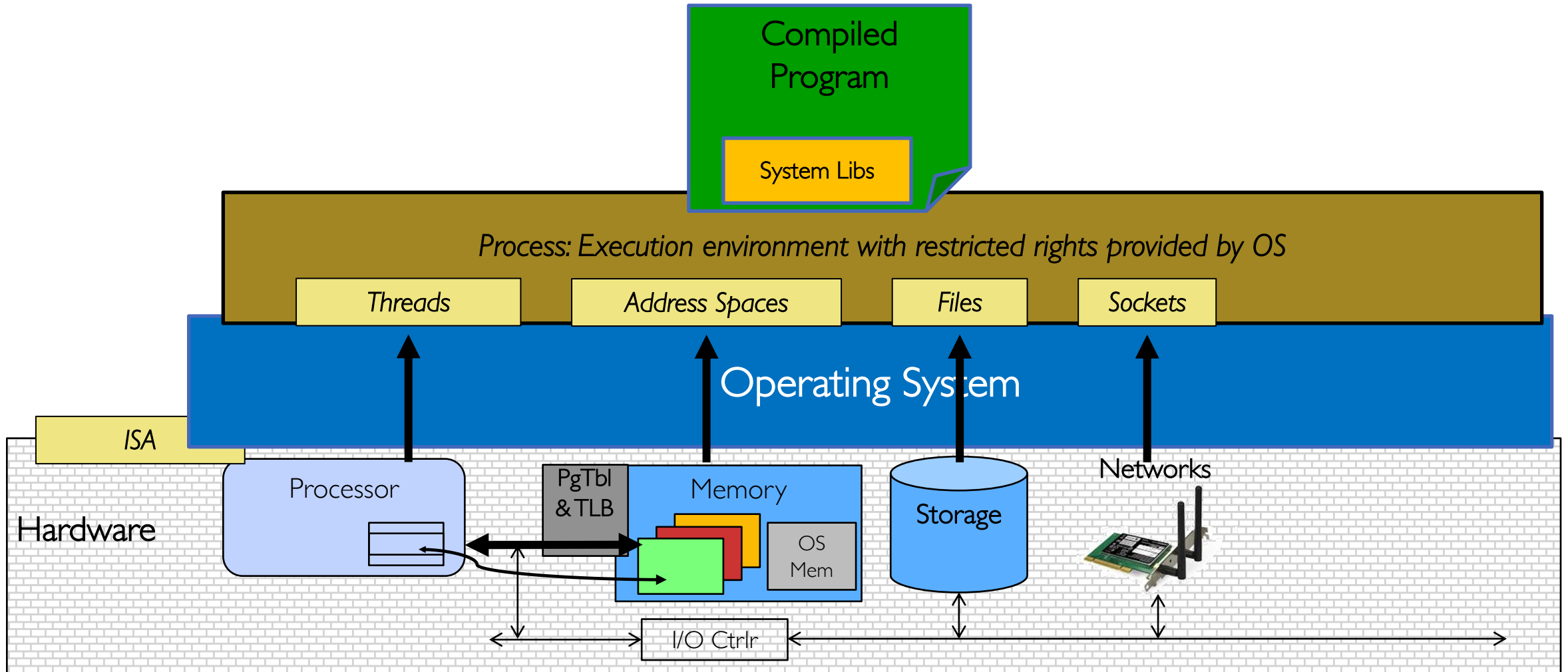
The OS *abstracts* these hardware details from the application

What is an Operating System?

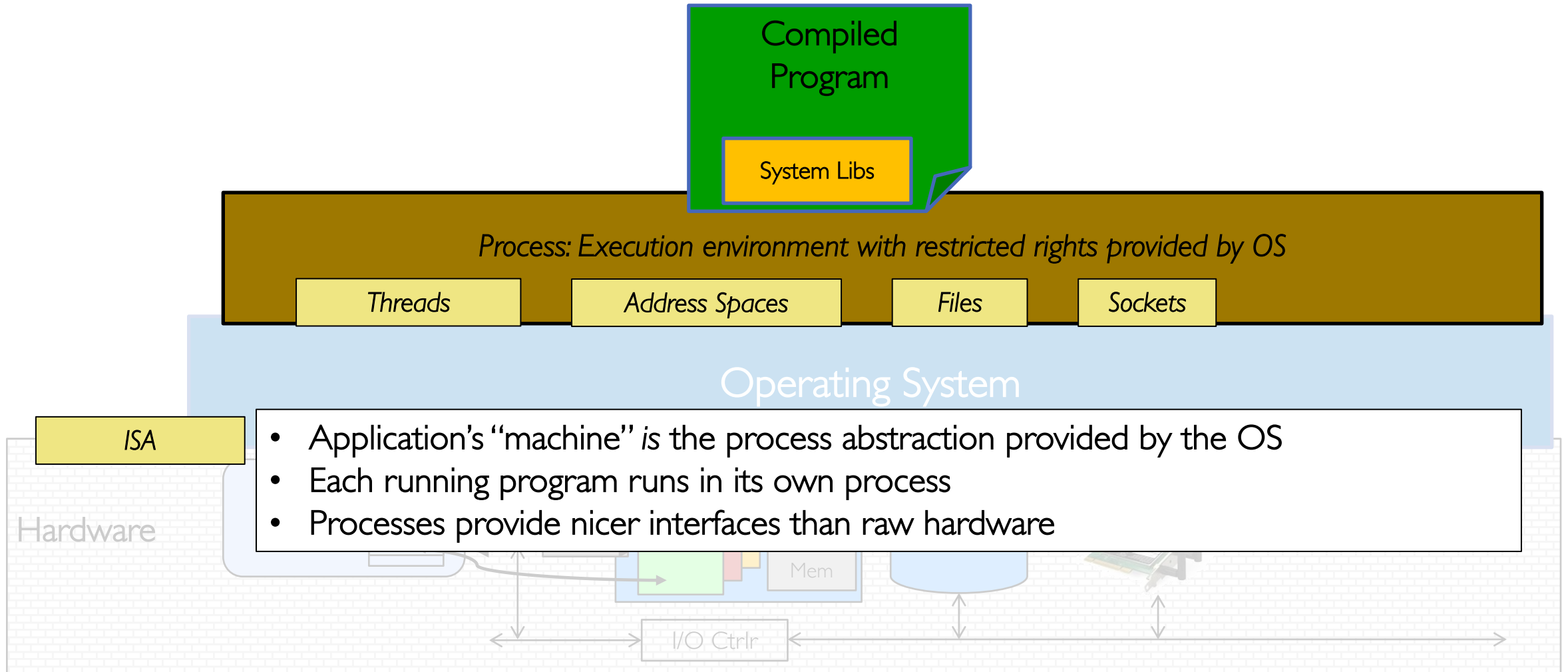


- Illusionist
 - Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

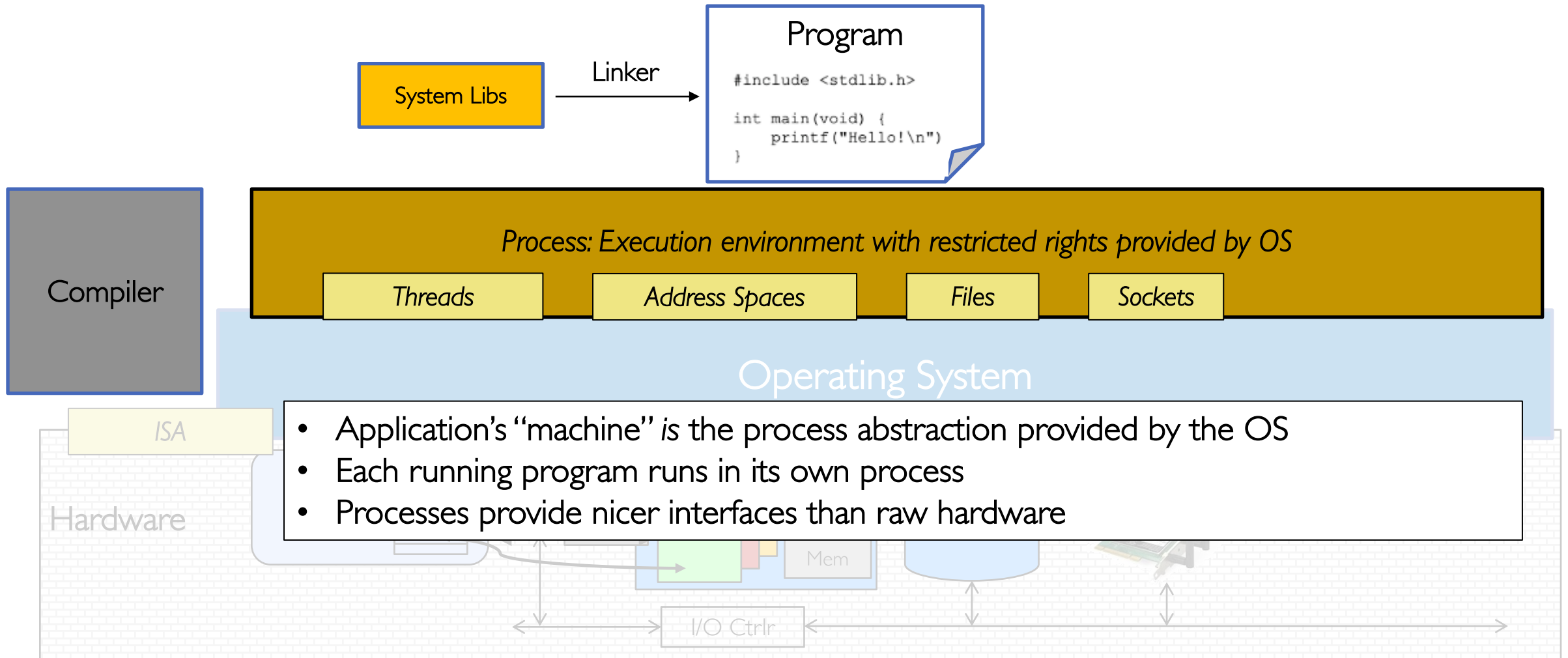
OS Basics: Virtualizing the Machine



Compiled Program's View of the World



System Programmer's View of the World

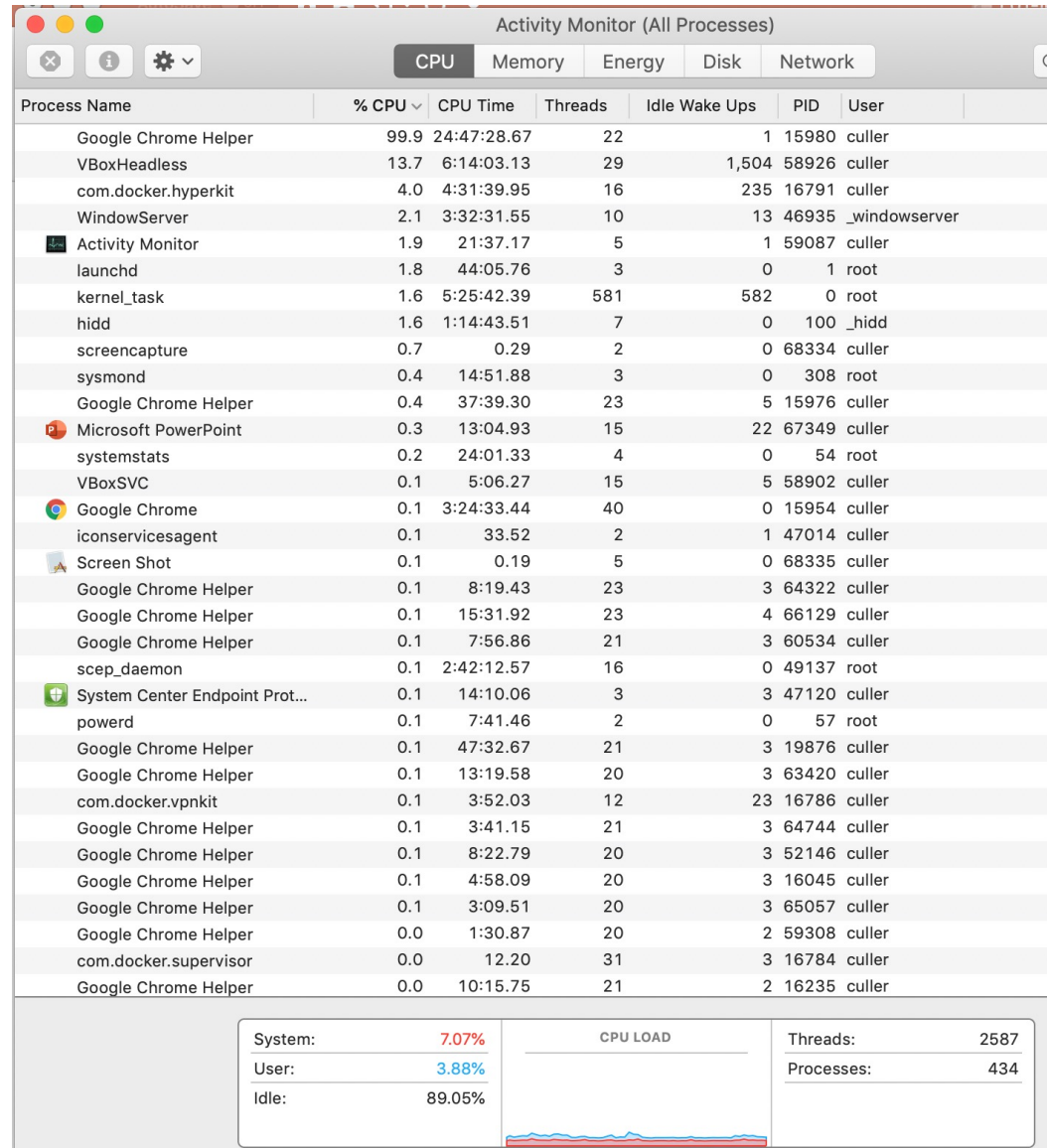


What's in a Process?

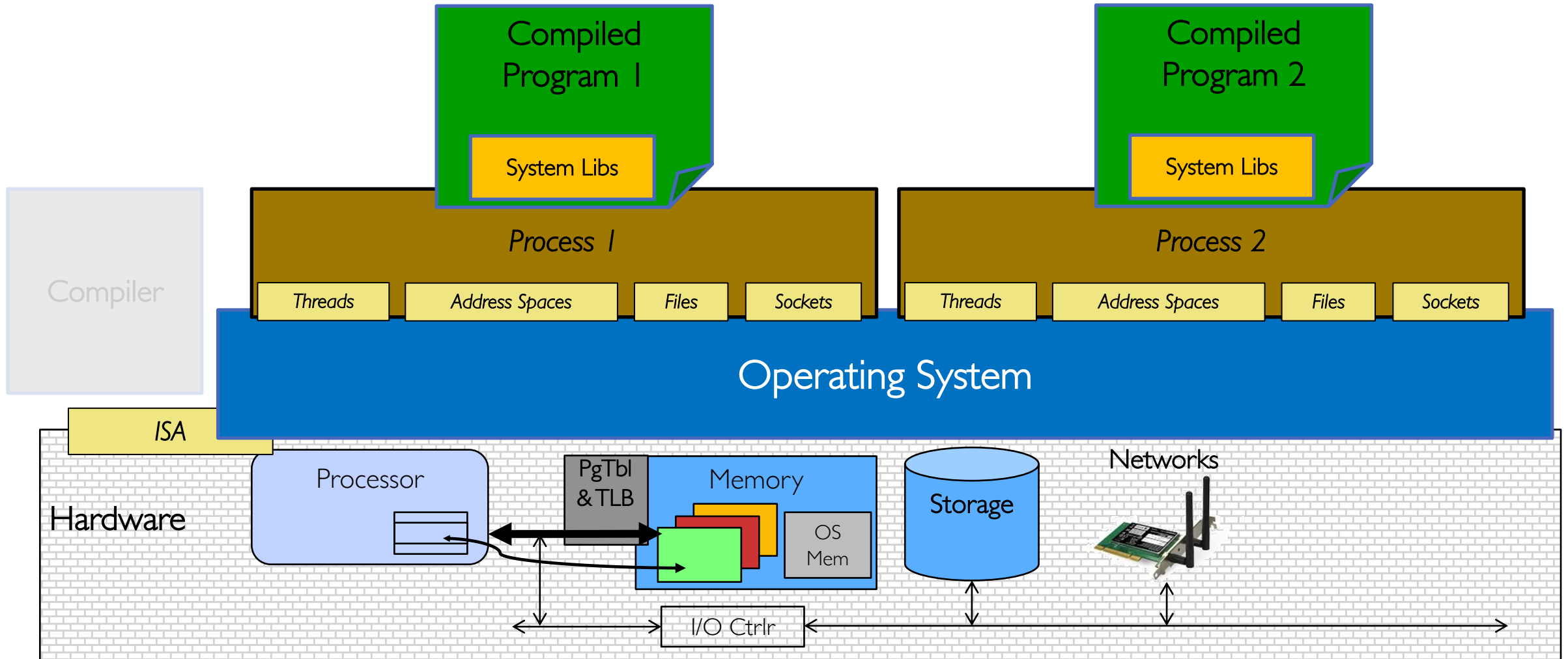
A process consists of:

- 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...

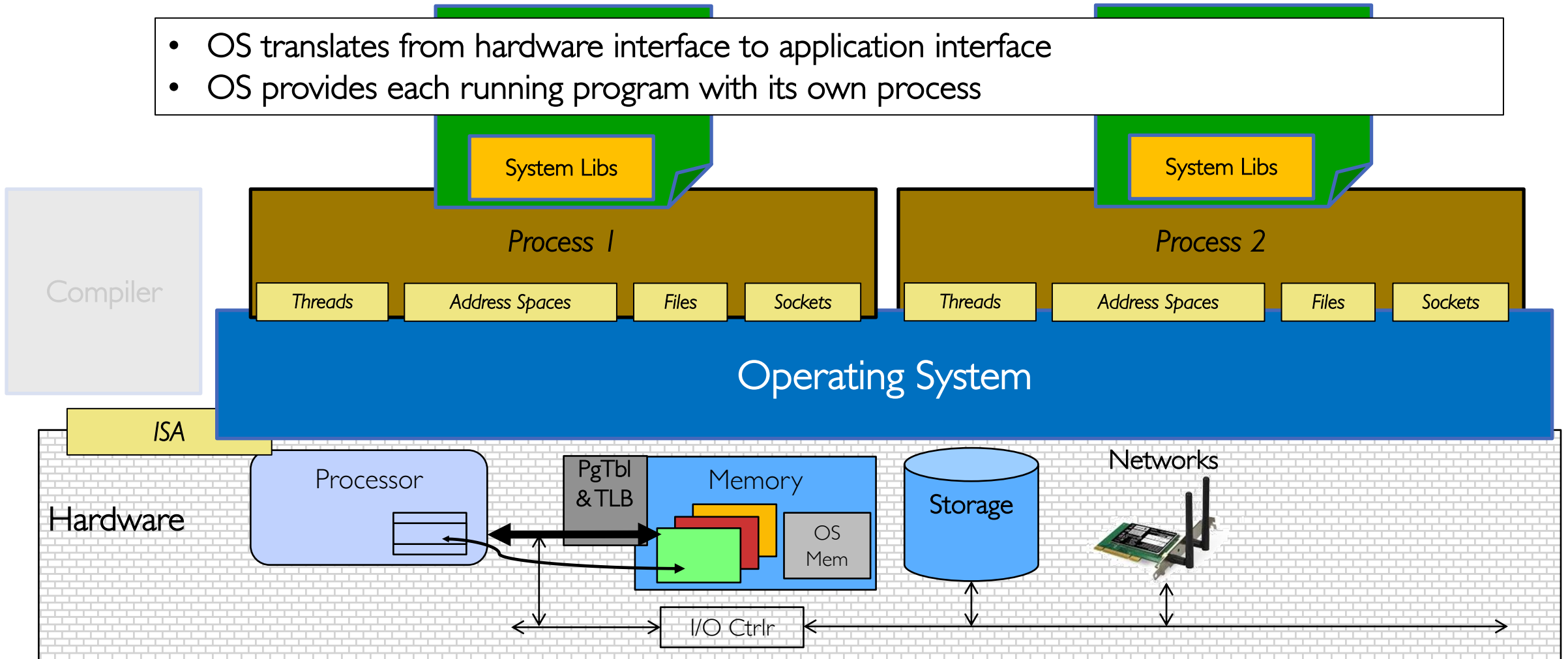


Operating System's View of the World



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

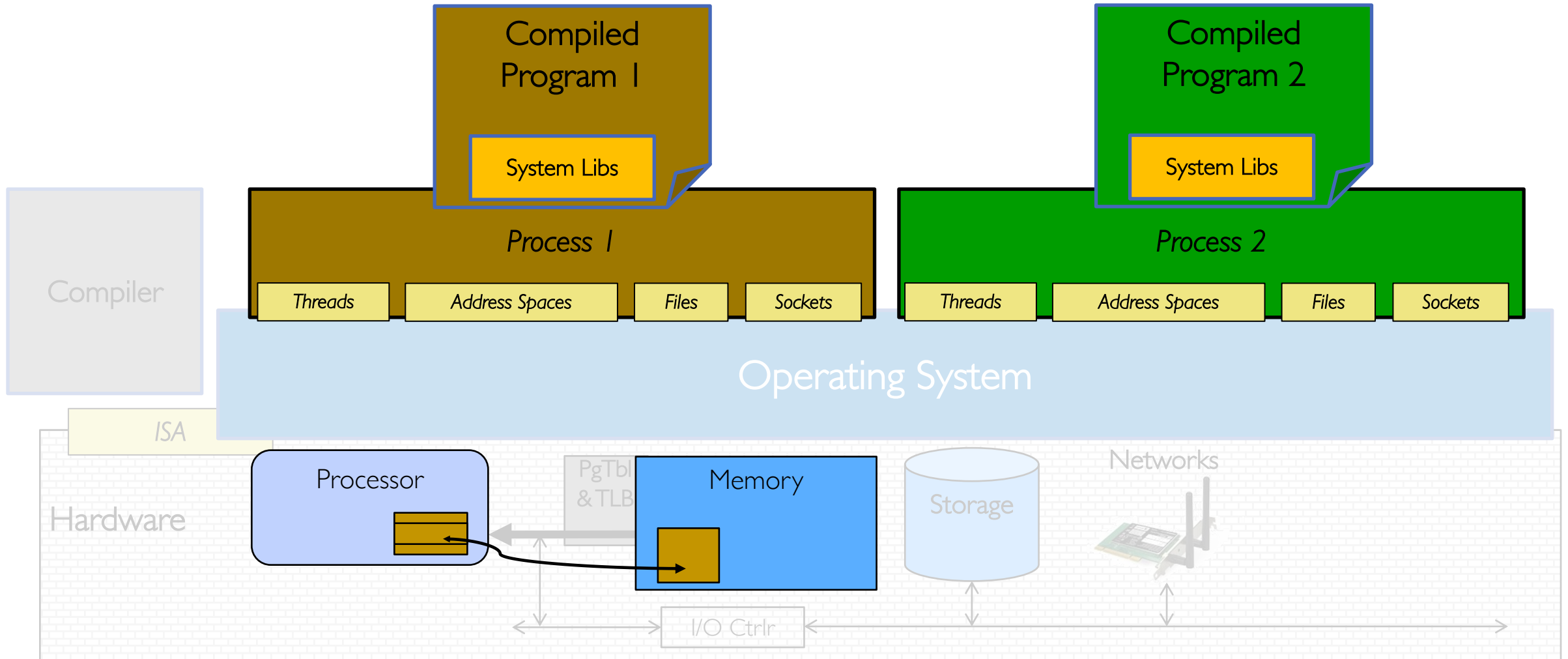
- Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication

- Illusionist

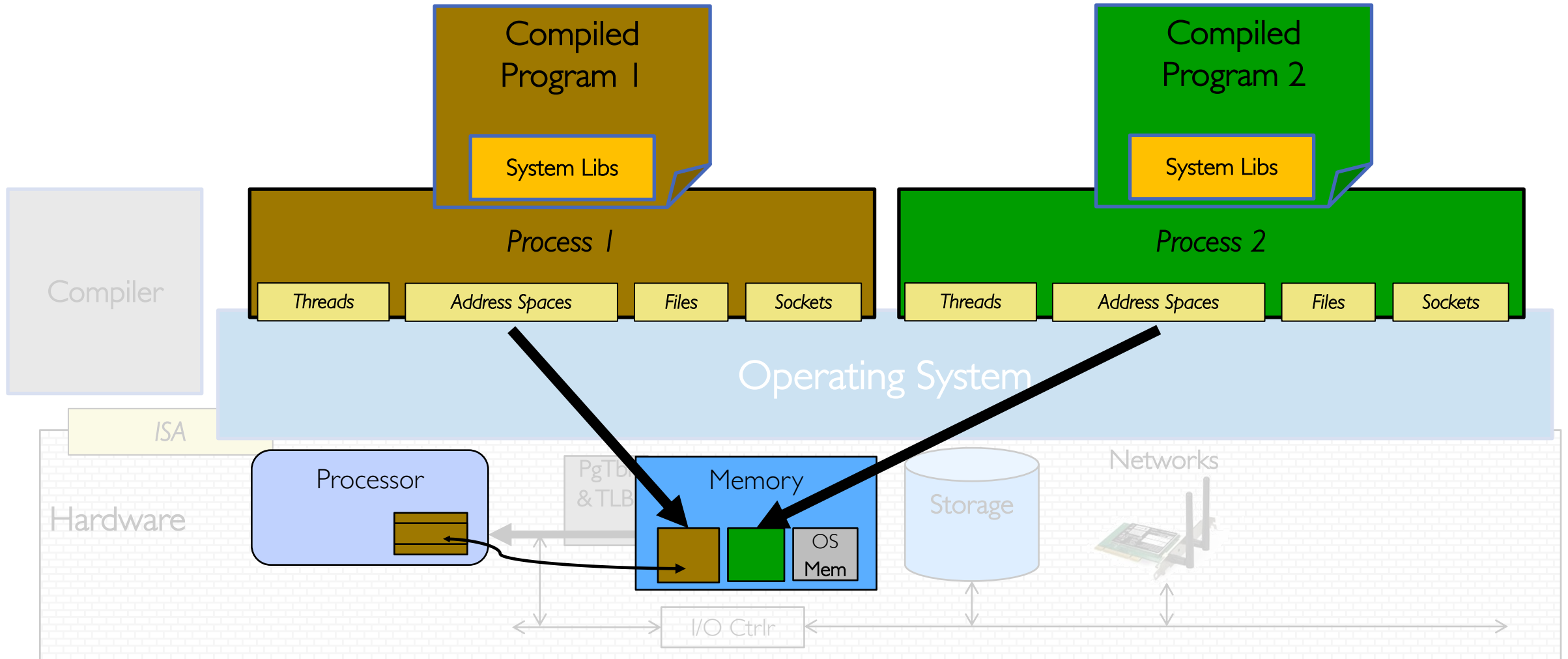
- Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization



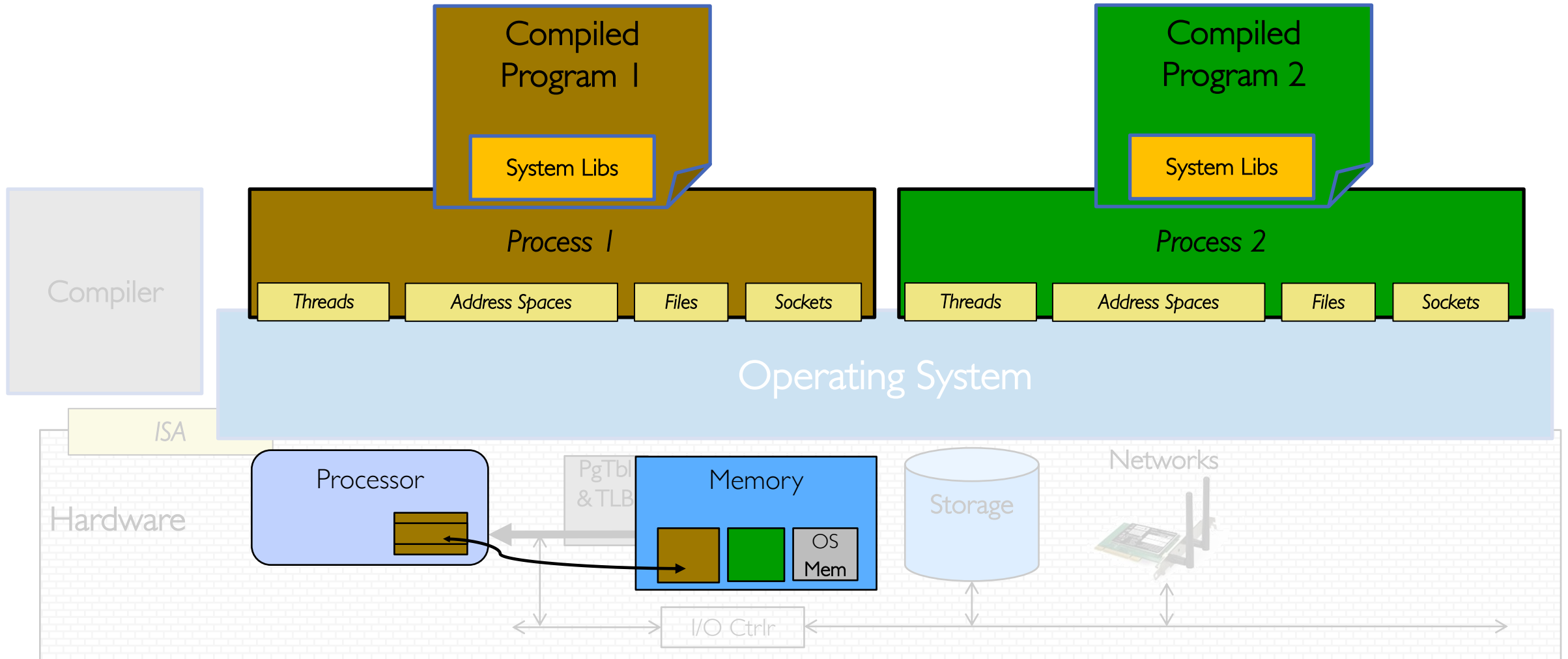
OS Basics: Running a Process



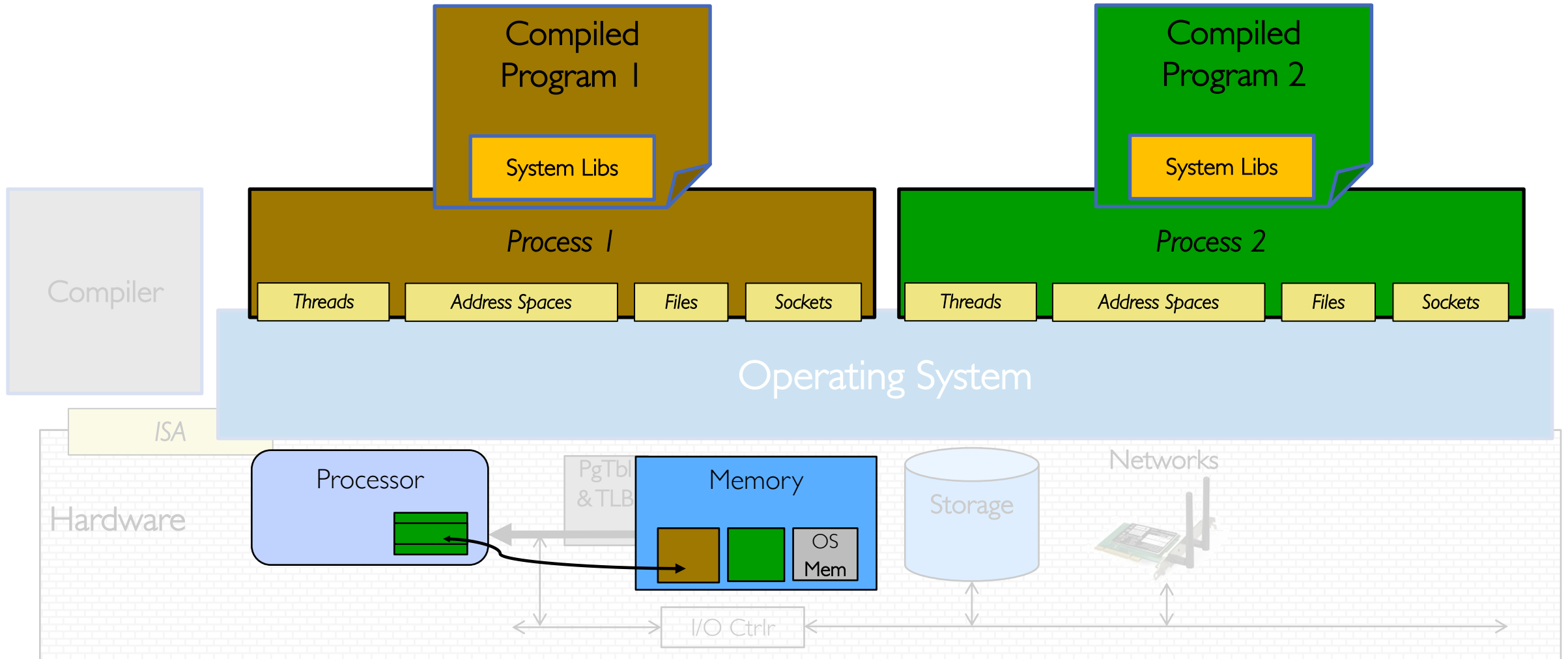
OS Basics: Switching Processes



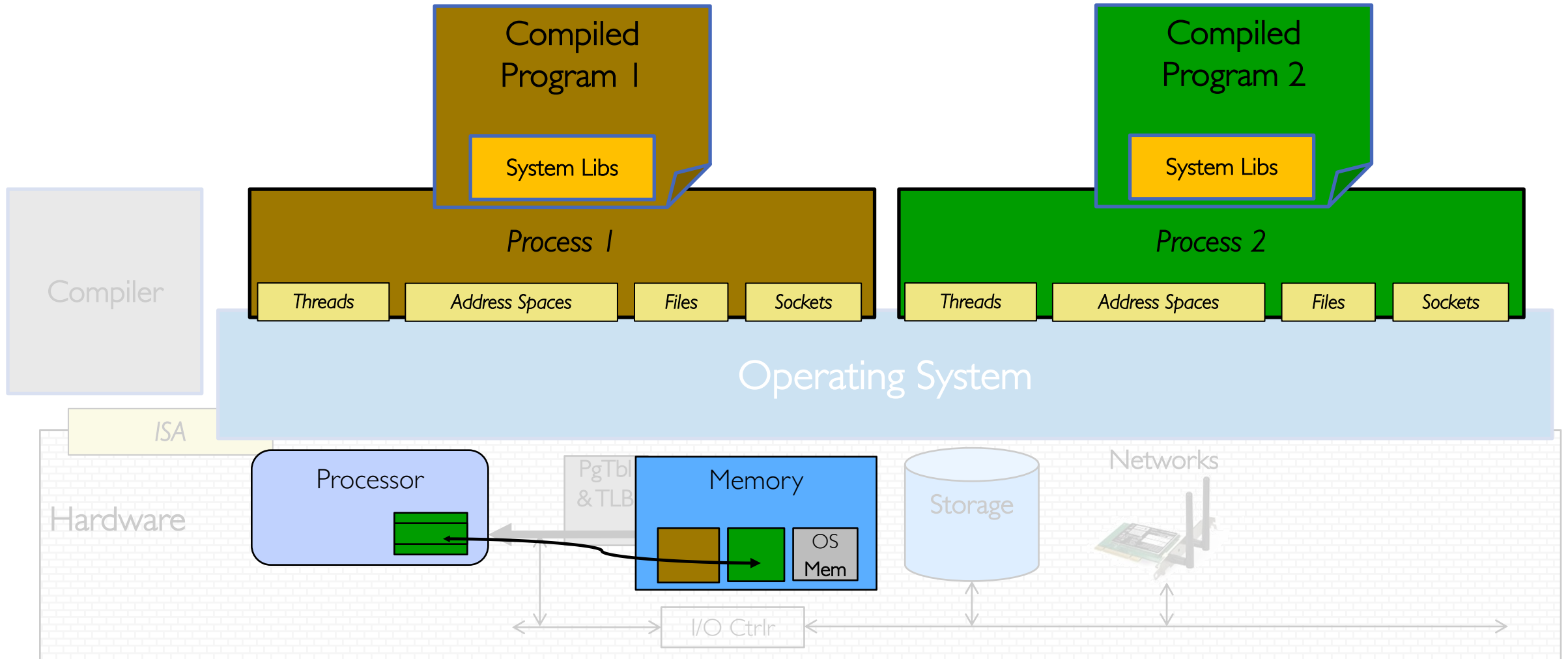
OS Basics: Switching Processes



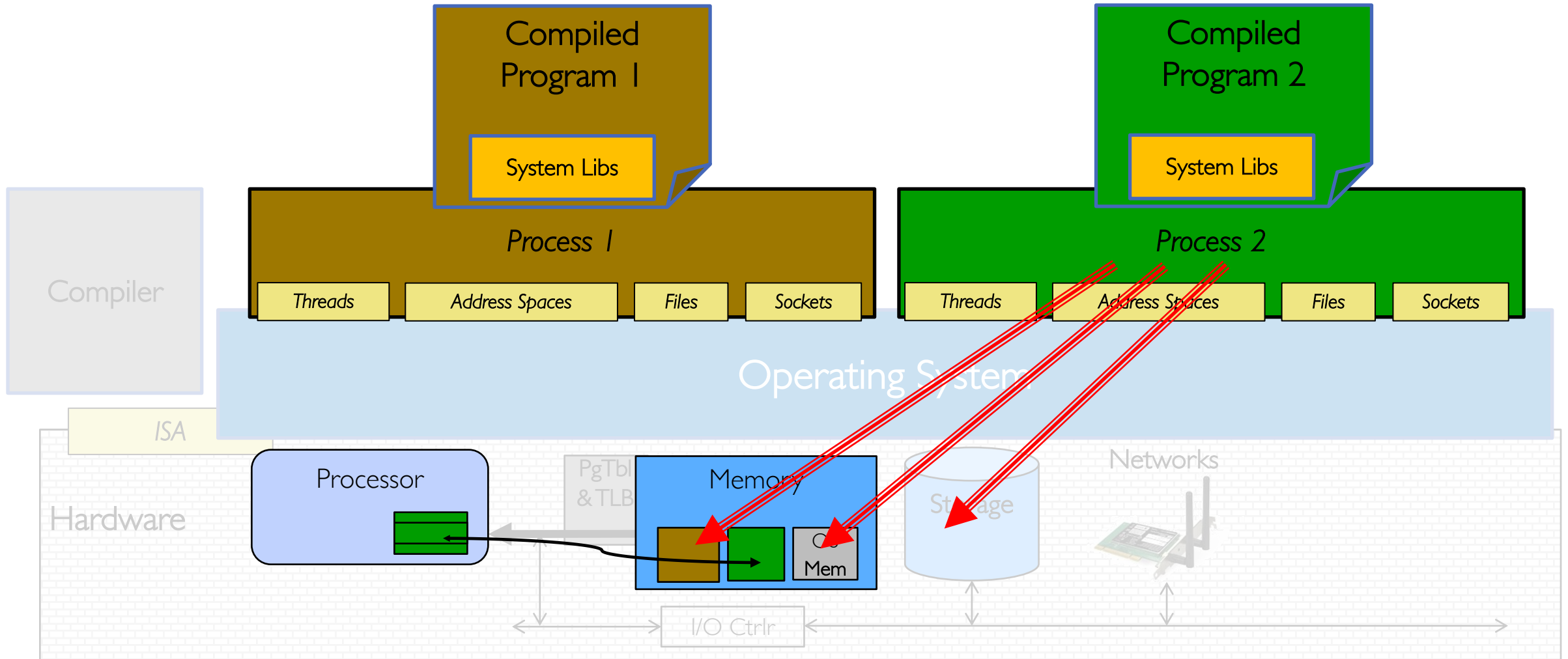
OS Basics: Switching Processes



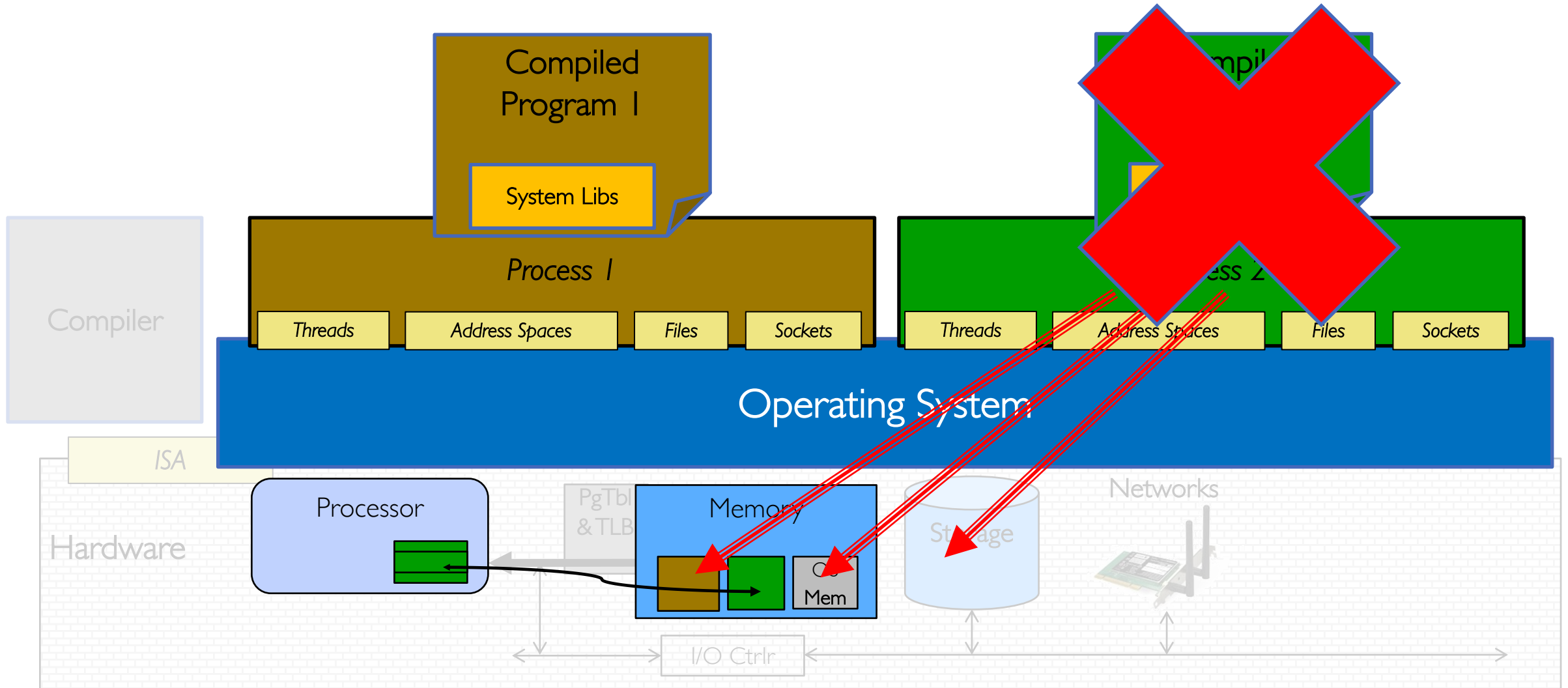
OS Basics: Switching Processes



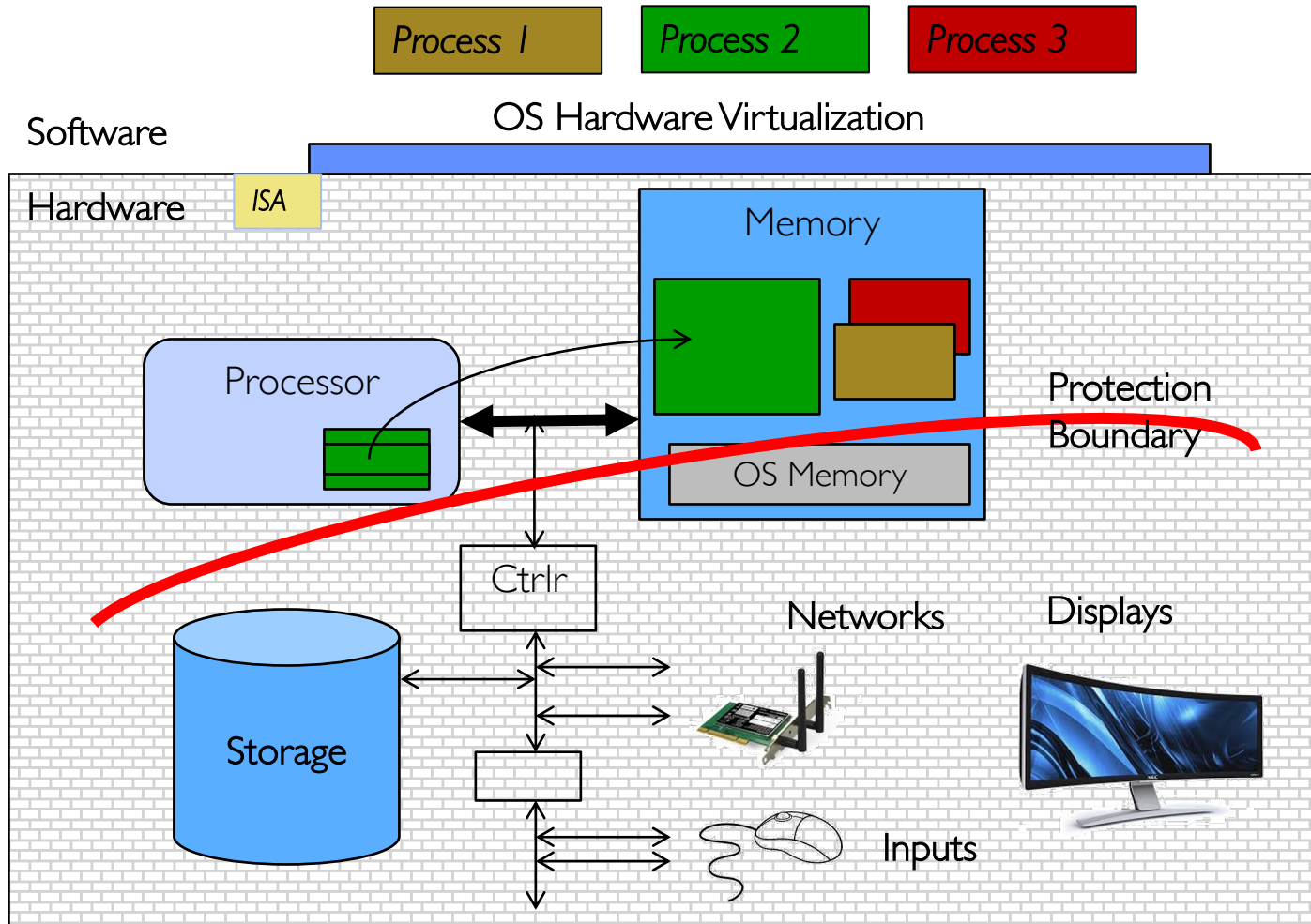
OS Basics: Protection



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

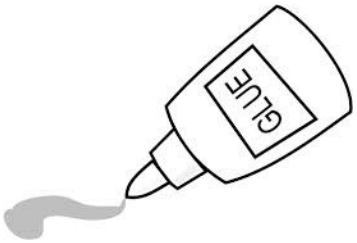
- Illusionist

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

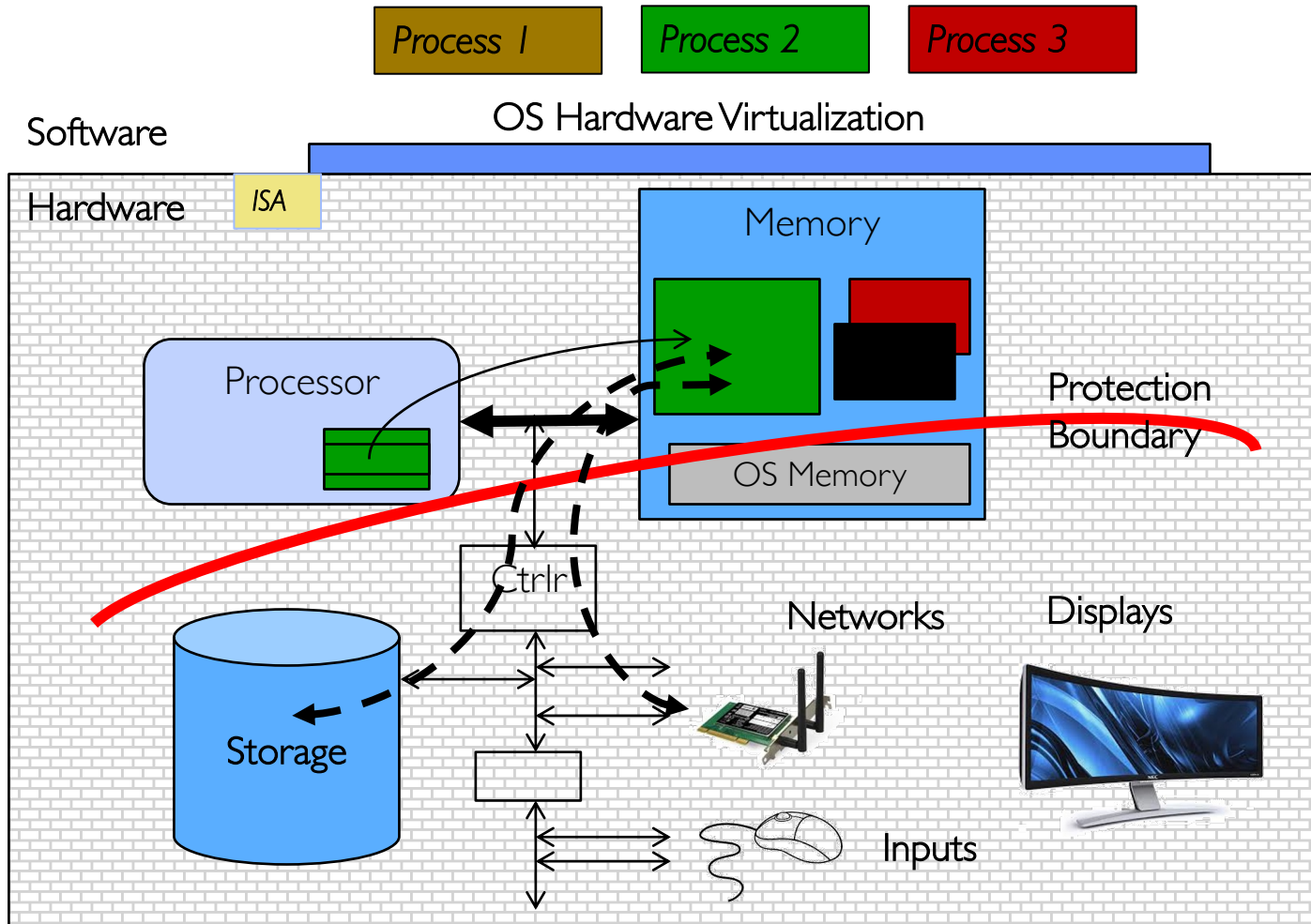


- Glue

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

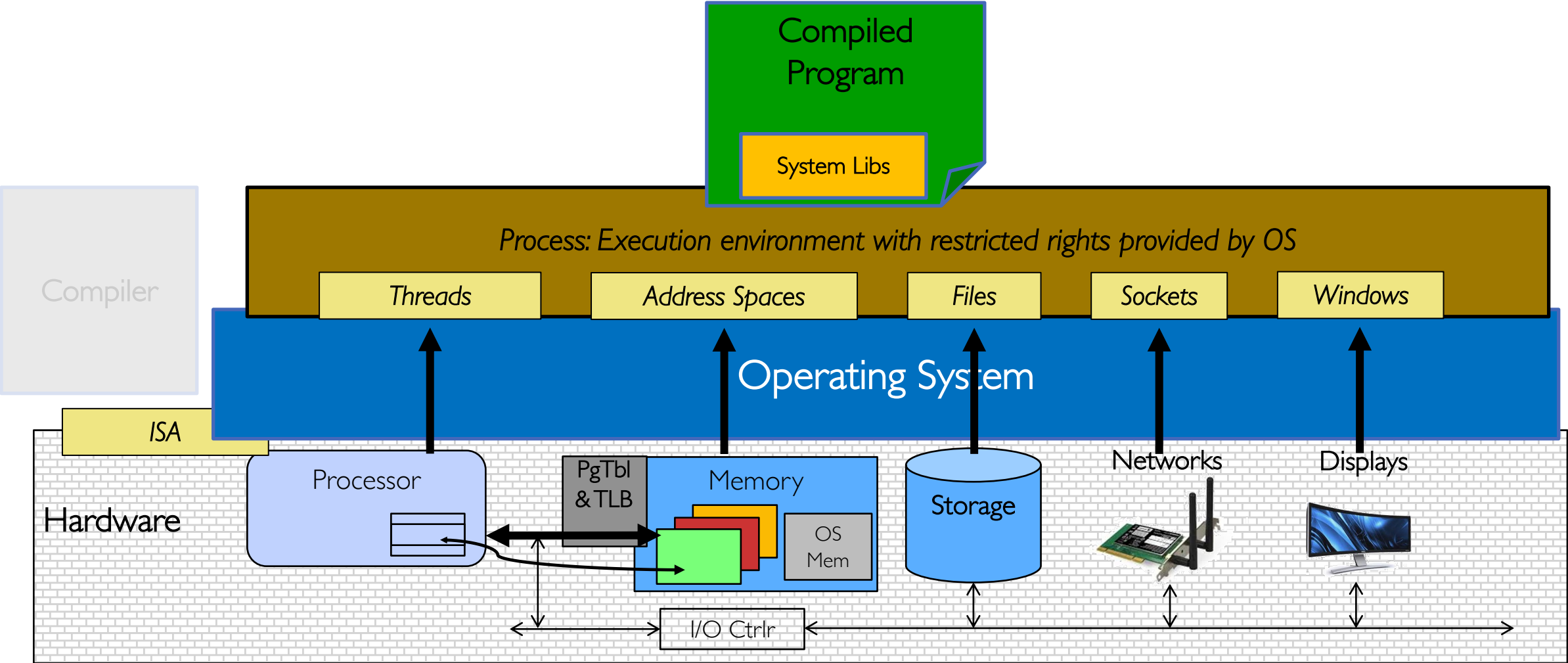


OS Basics: I/O

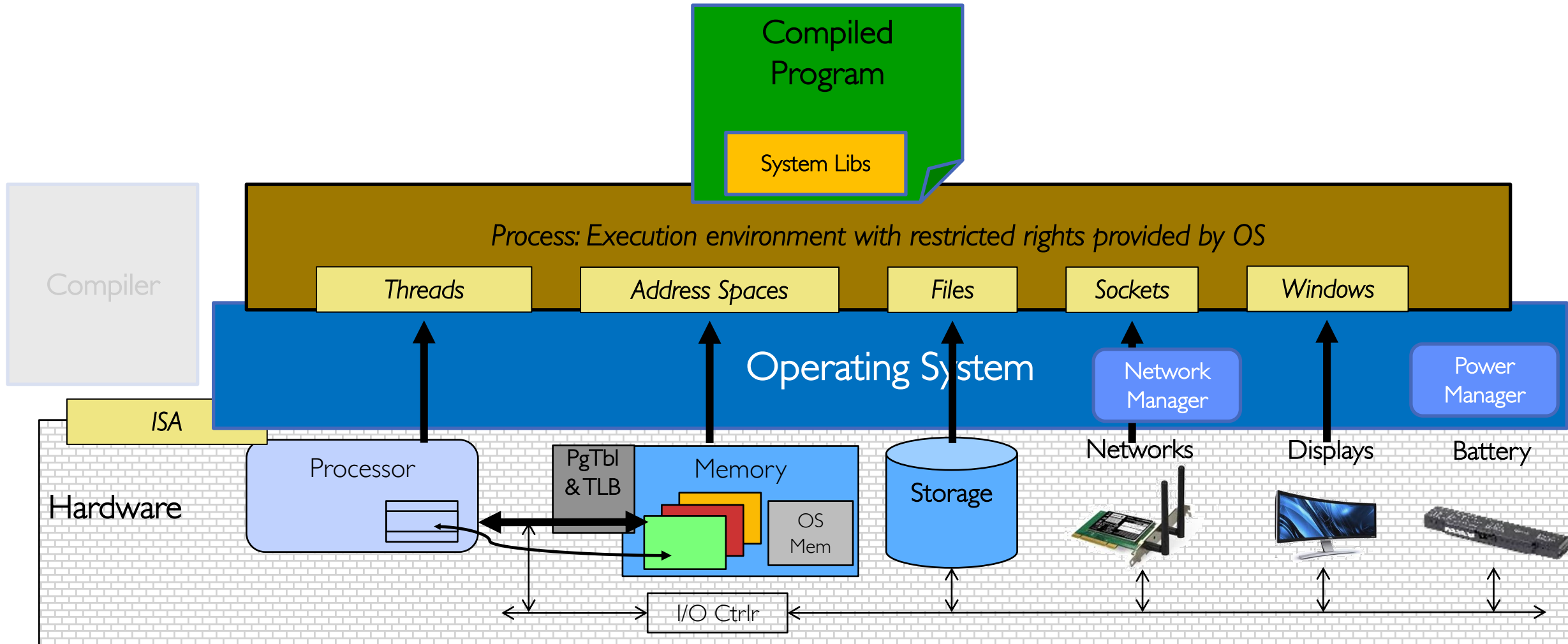


- OS provides common services in the form of I/O

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

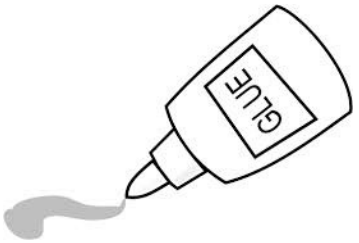
- Illusionist

- 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

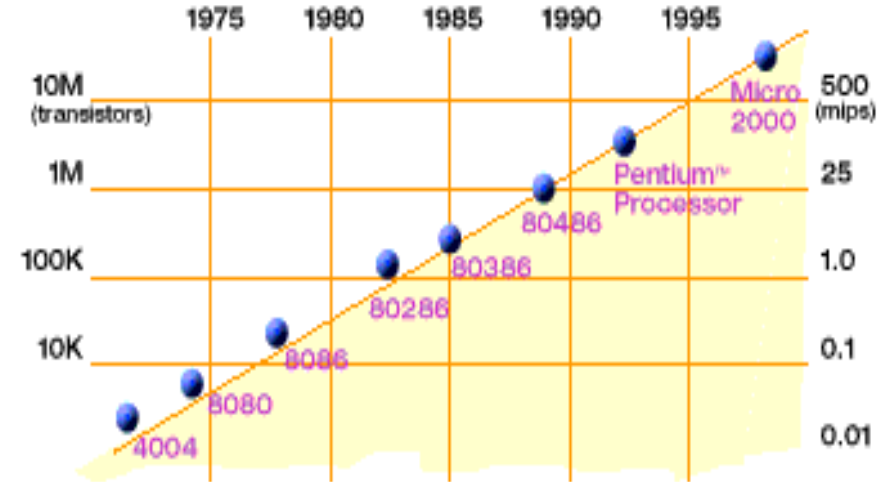
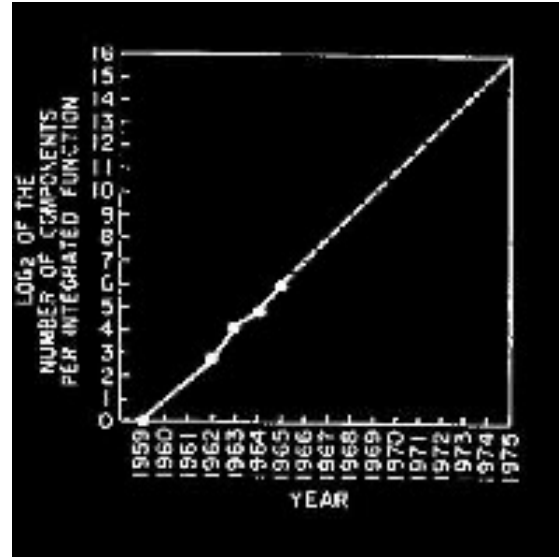
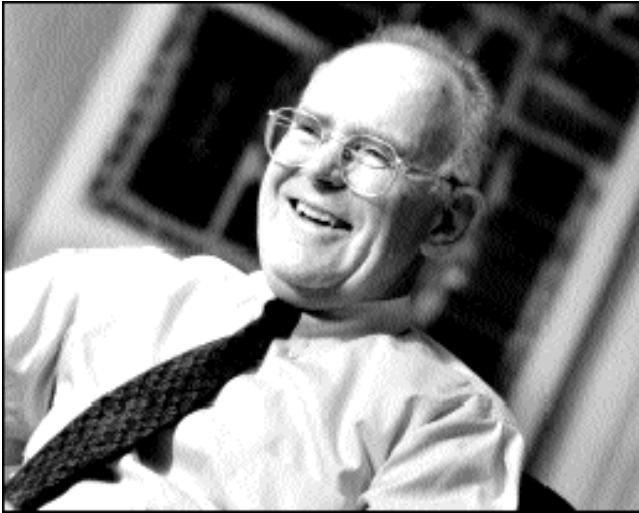


Why take Operating Systems?

- 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"

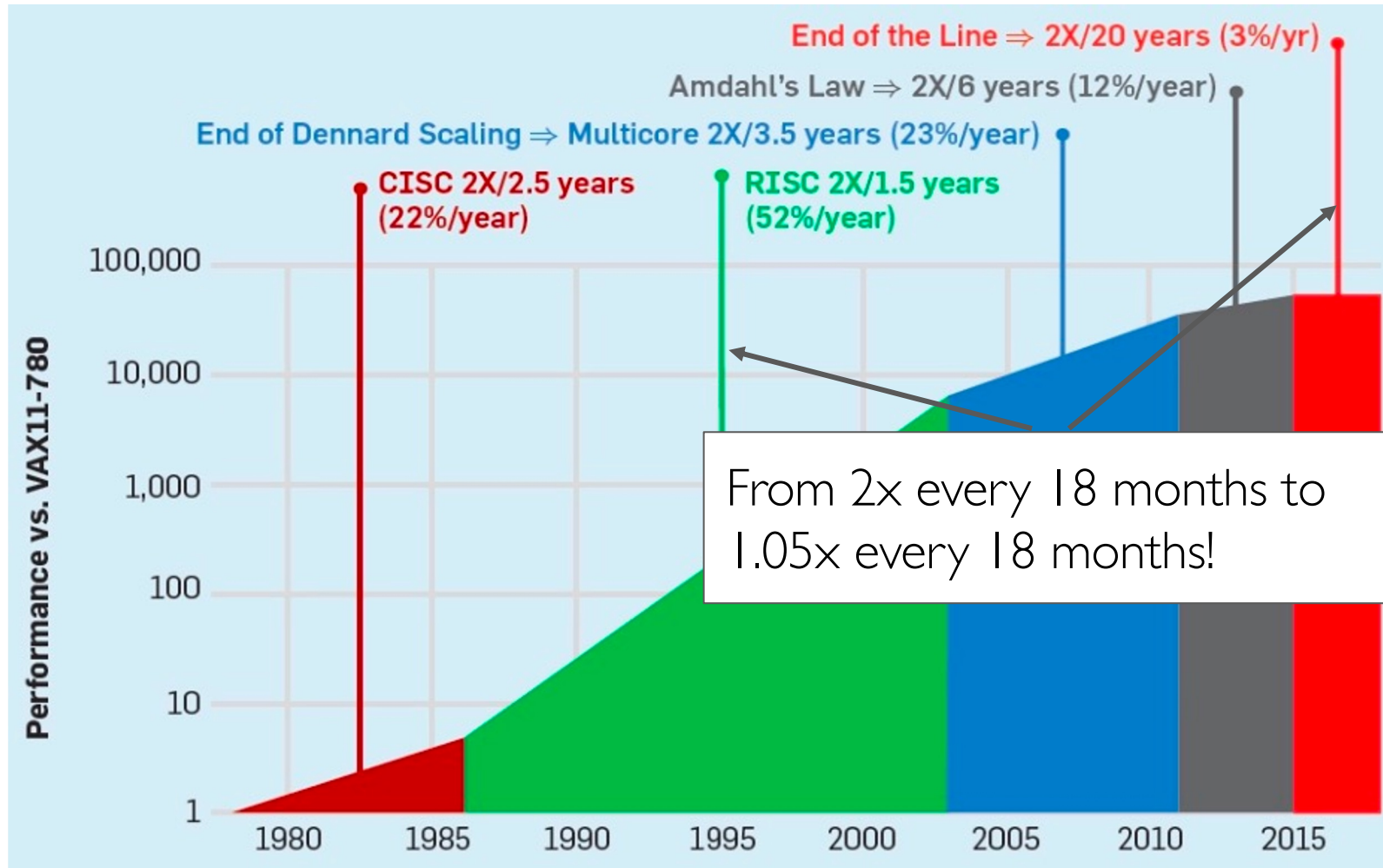
Gordon Moore (co-founder of Intel) predicted in 1965 that the **transistor density** of semiconductor chips **would double roughly every 2 years**

- Microprocessors have become smaller, denser, and more powerful

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

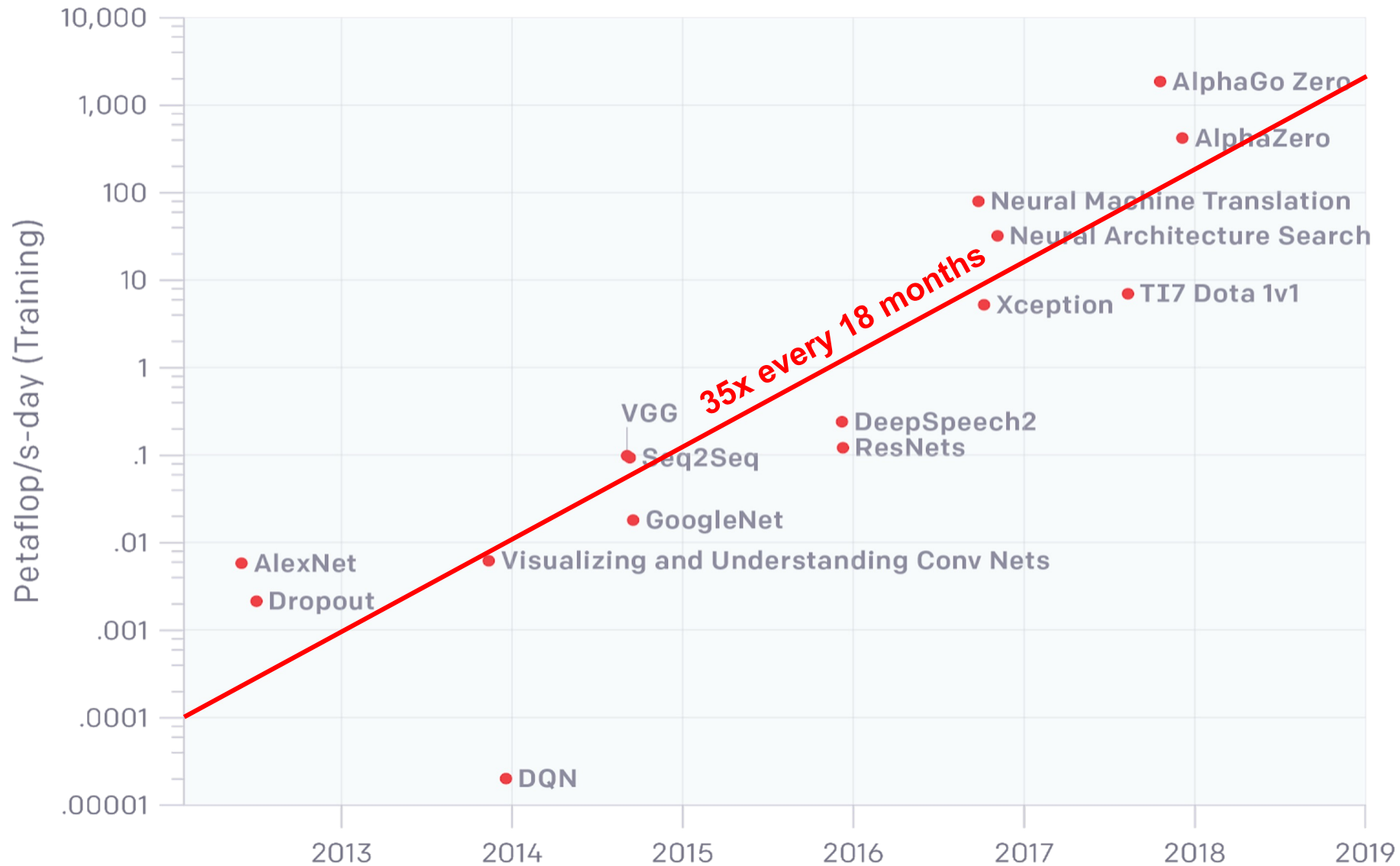
- Faster growth because transistors are closer to each other so electrical signals travel faster

The end of the Moore's Law



AI Compute demands

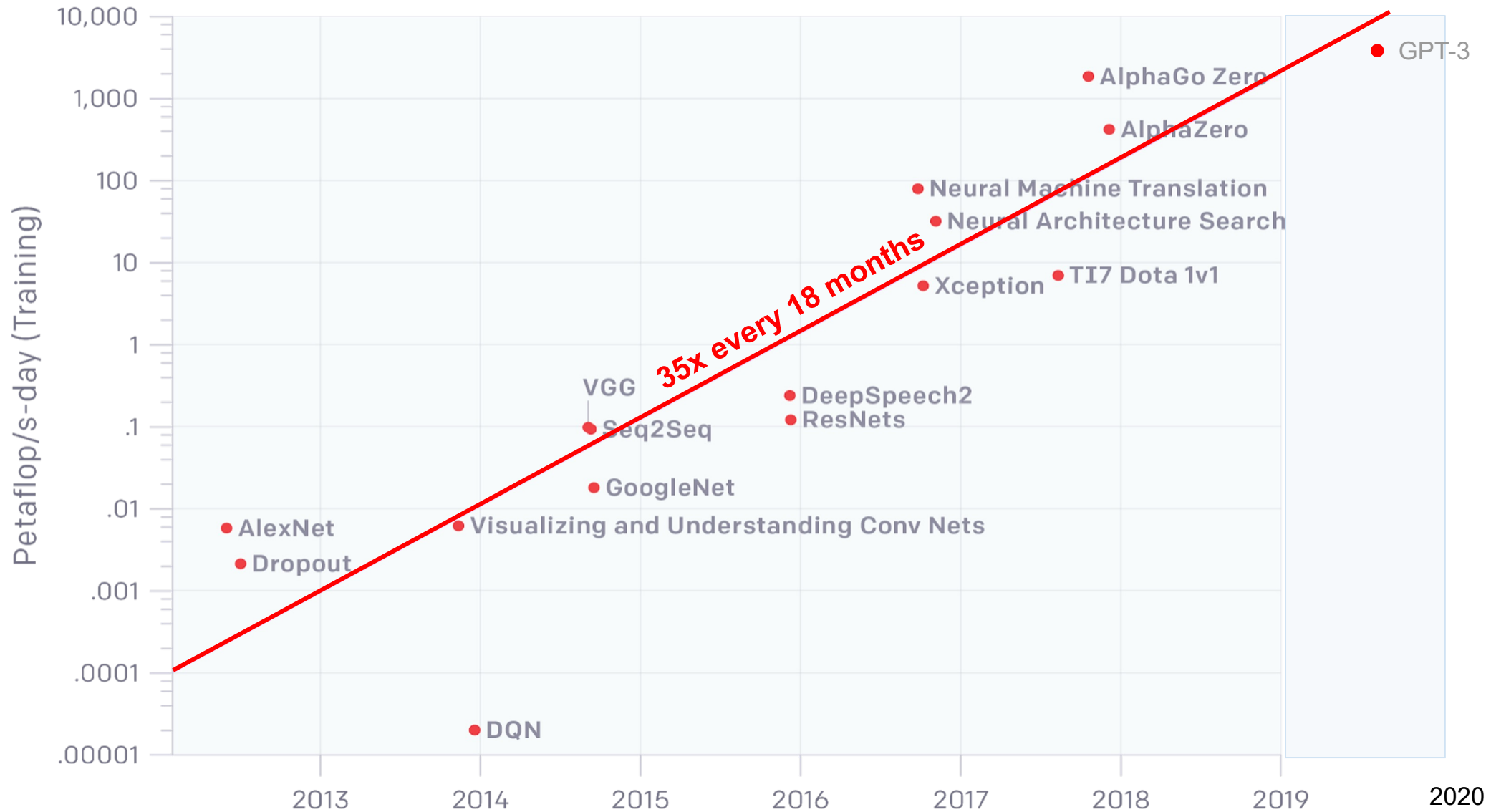
AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



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

AI Compute demands

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

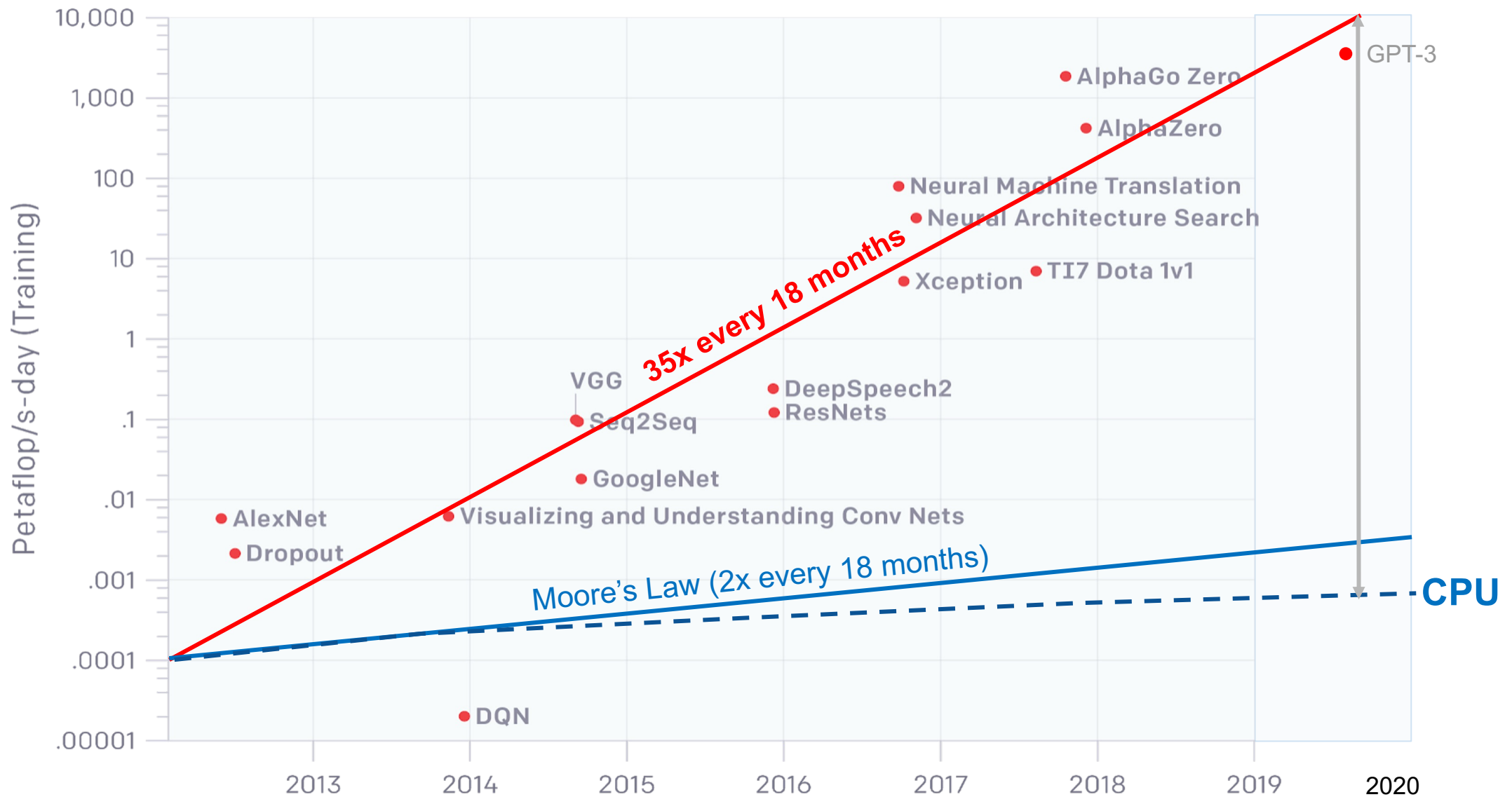


(<https://openai.com/blog/ai-and-compute/>)

Growing gap between demand and supply

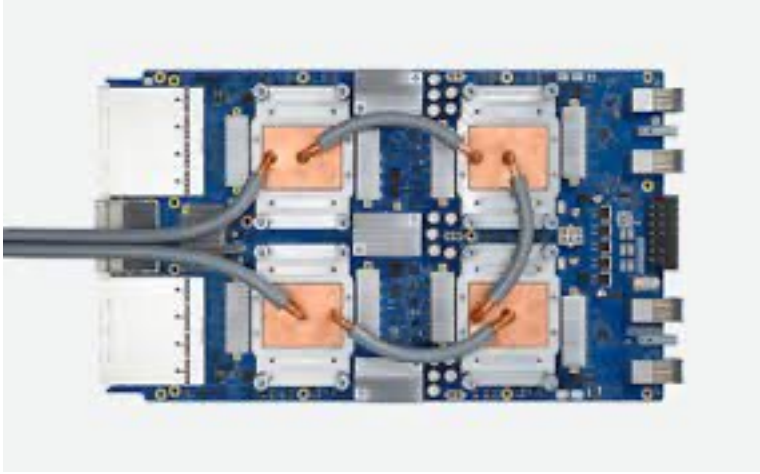
AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

GPT-4 and beyond



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

Specialized processors to the rescue?



TPU v3



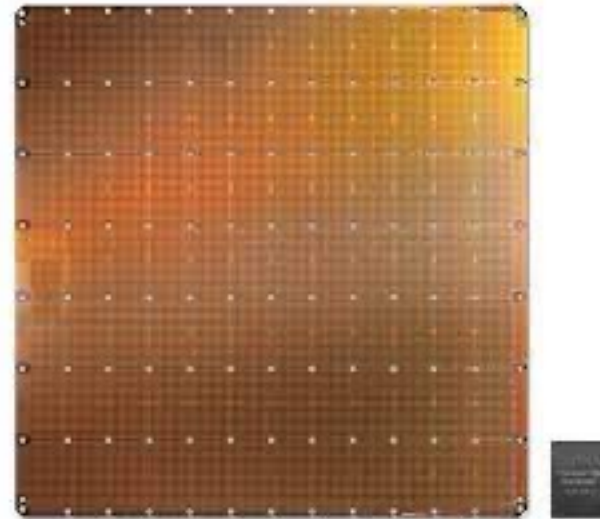
Nvidia A100



AWS's Inferentia



AMD Radeon



Cerebras

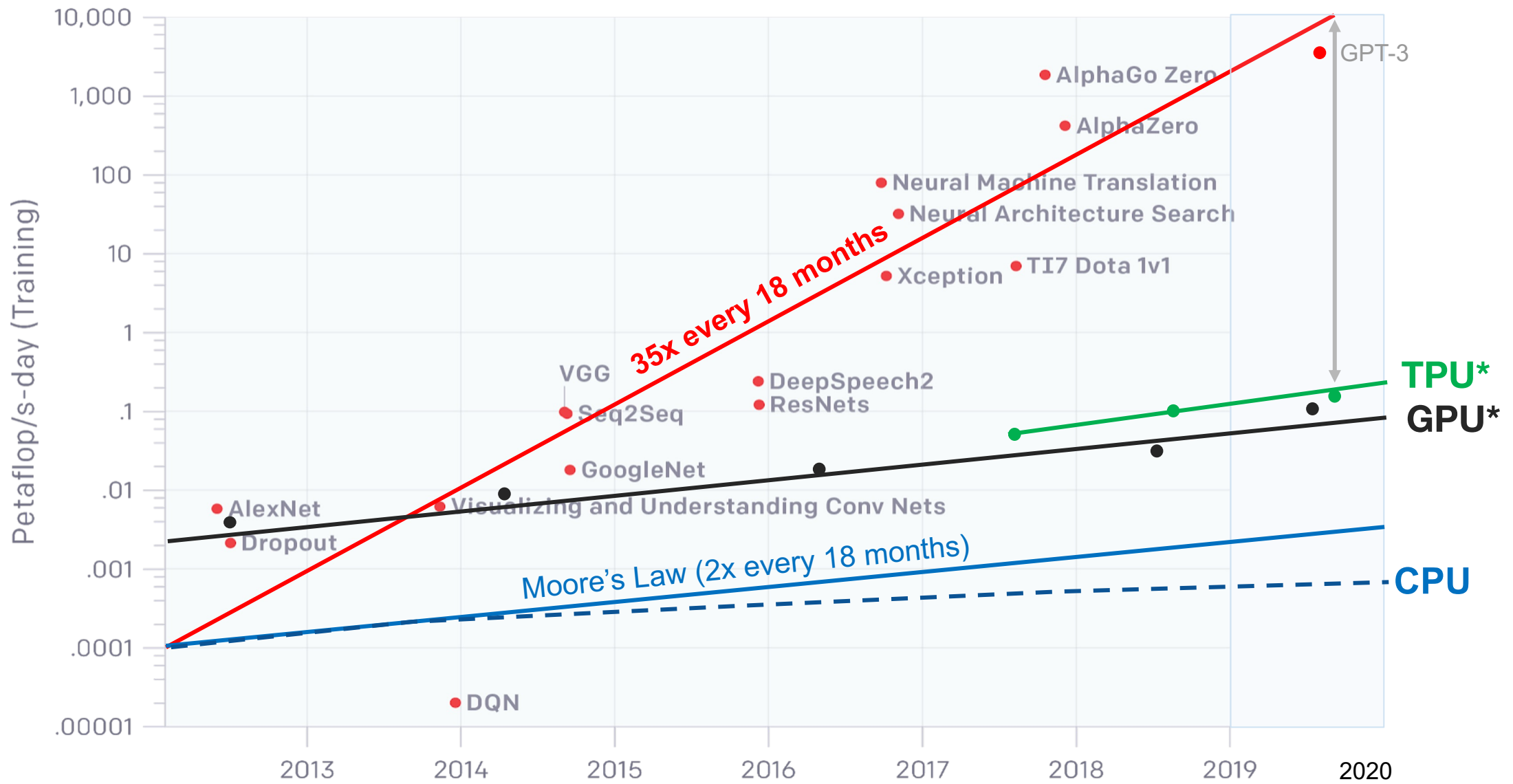


Intel Alchemist

Specialized hardware not enough

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

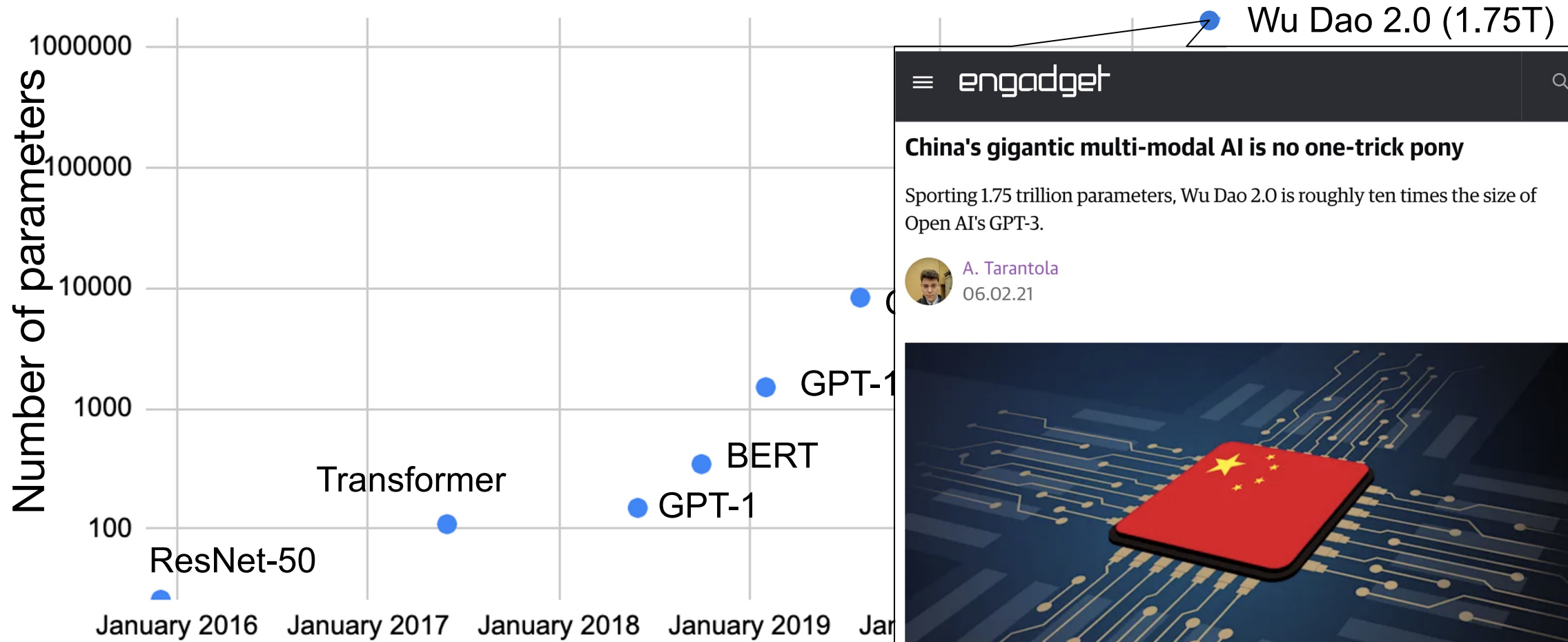
GPT-4 and beyond



* assume 0.33 utilization

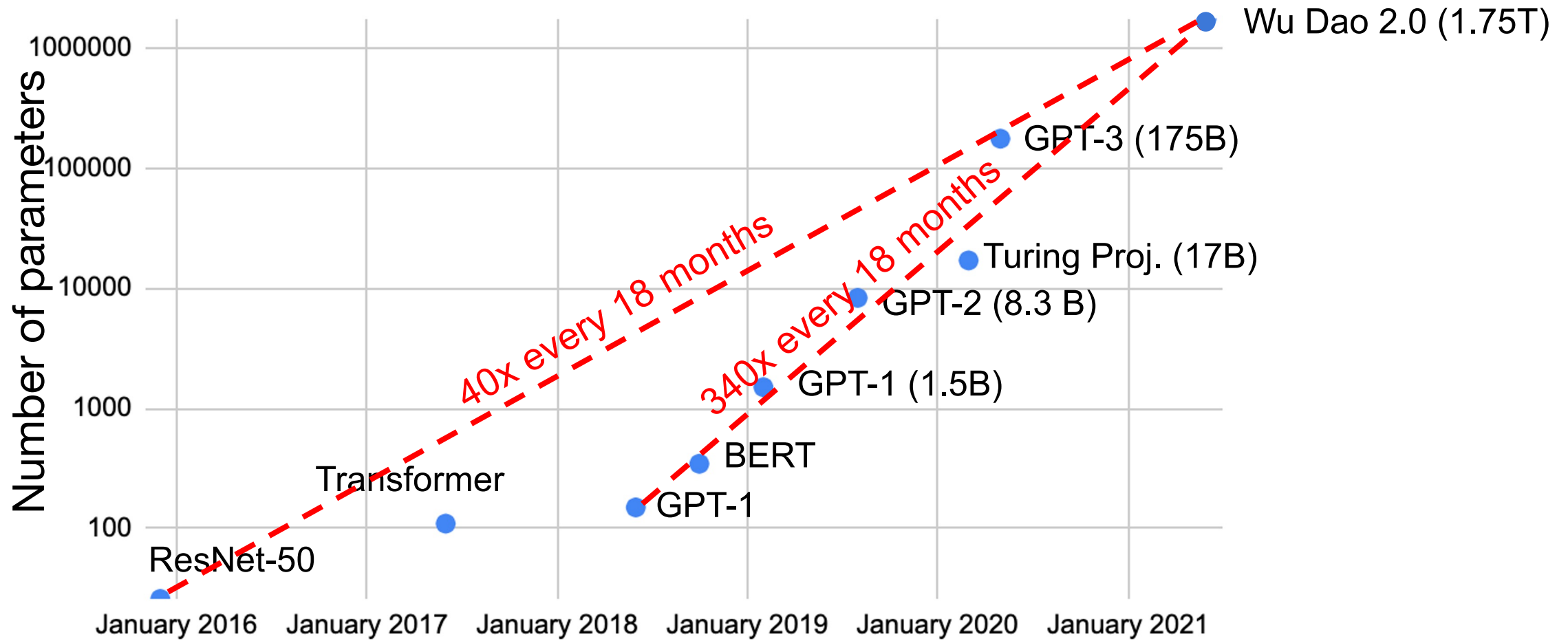
<https://openai.com/blog/ai-and-compute/>

Memory demands growing as fast

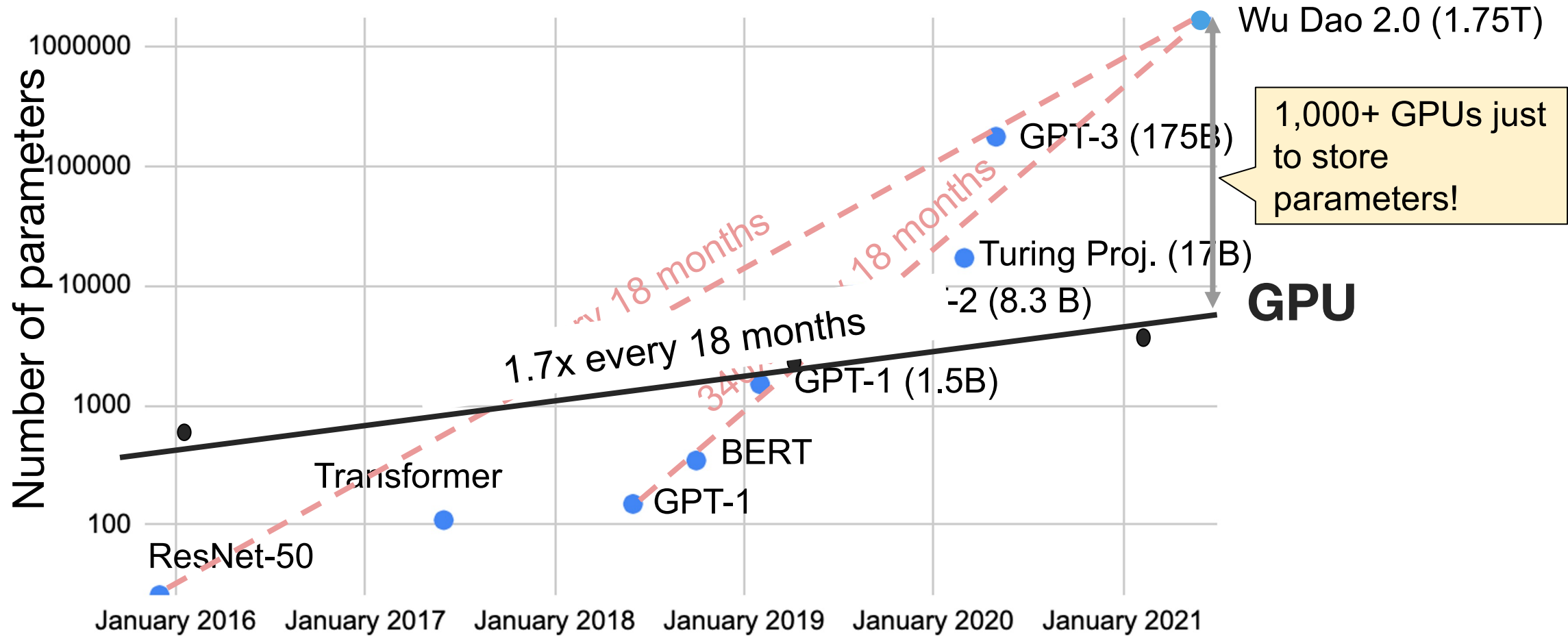


The screenshot shows a mobile article from Engadget. The title is "China's gigantic multi-modal AI is no one-trick pony". The text states: "Sporting 1.75 trillion parameters, Wu Dao 2.0 is roughly ten times the size of Open AI's GPT-3." The author is A. Tarantola, and the date is 06.02.21. The image at the bottom of the article depicts a red square with the Chinese flag's stars, resting on a dark blue background with glowing yellow circuit traces.

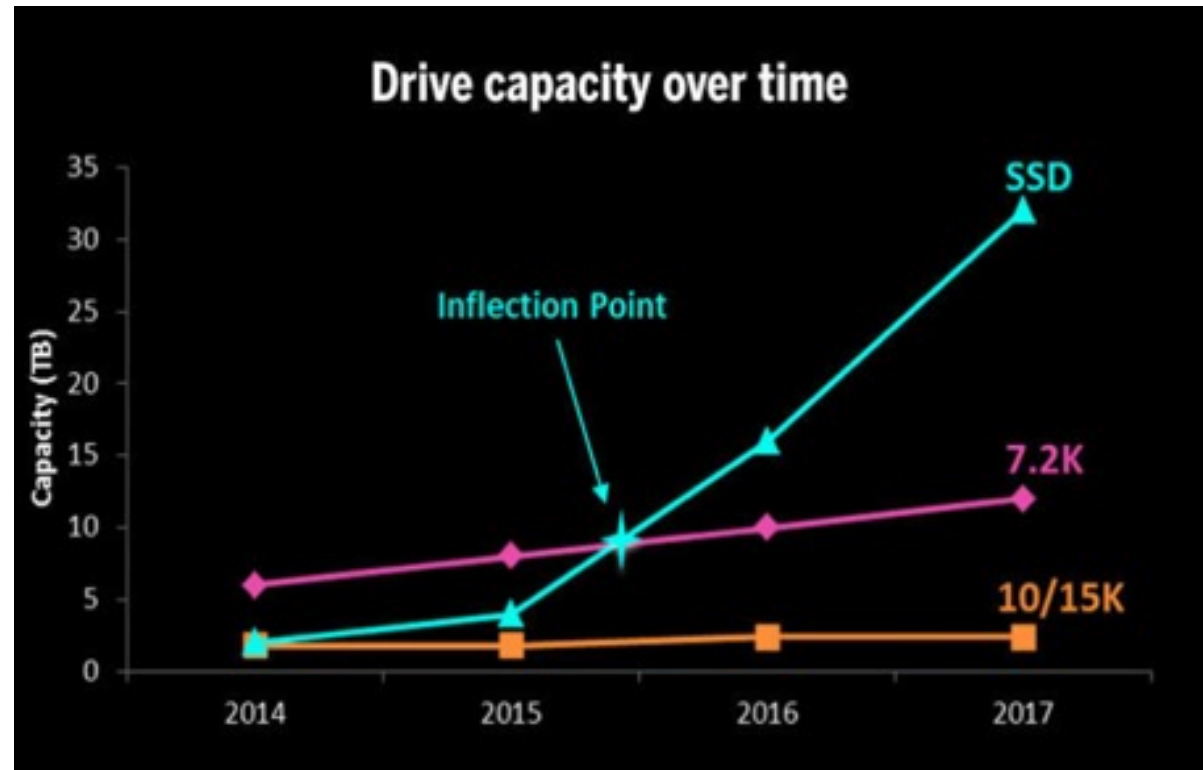
Memory demands growing as fast



Huge Gap between memory demands and single-chip memory



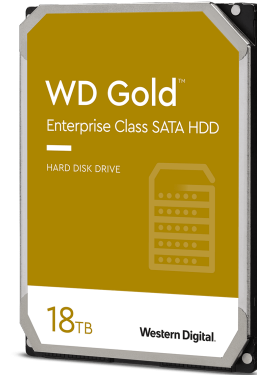
Storage Capacity is Still Growing!



Storage Capacity



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



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



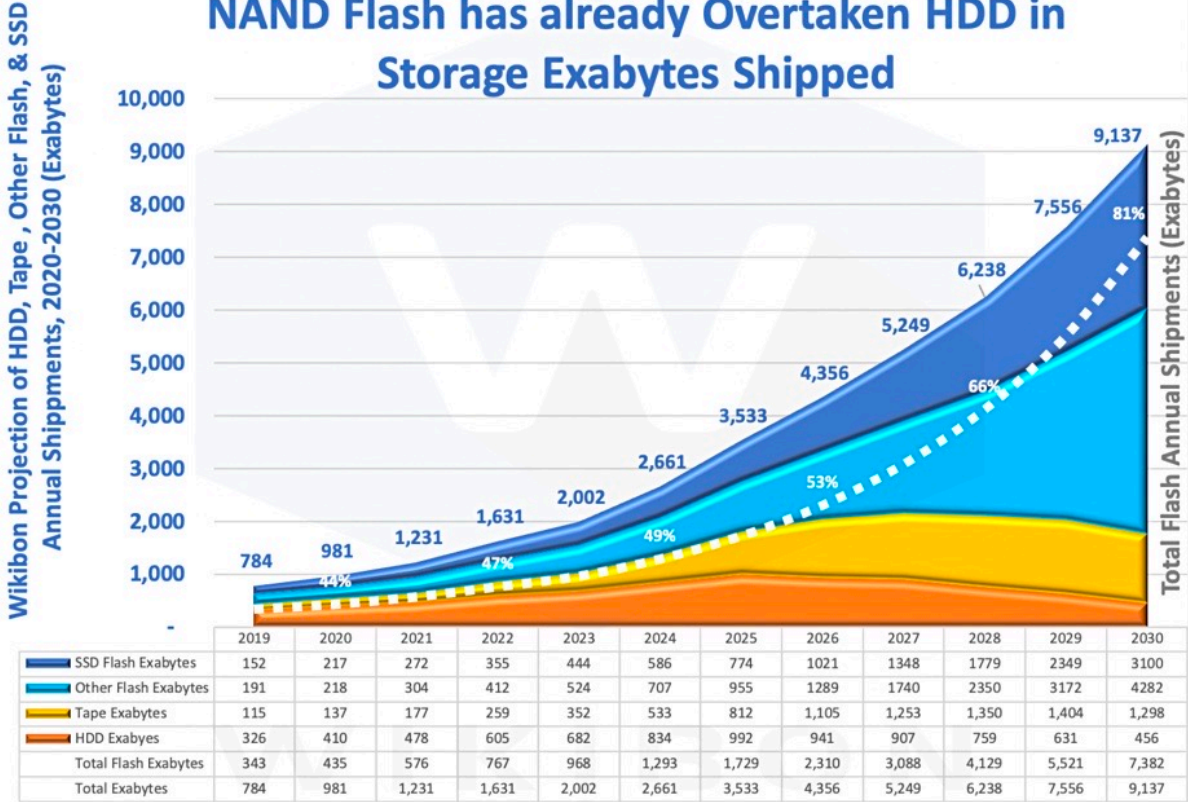
Samsung SSD 2.5-inch drive:
4 TB @ \$380K (\$95/TB)

Cheaper, but

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

SSD/Flash will Dominate

NAND Flash has already Overtaken HDD in Storage Exabytes Shipped

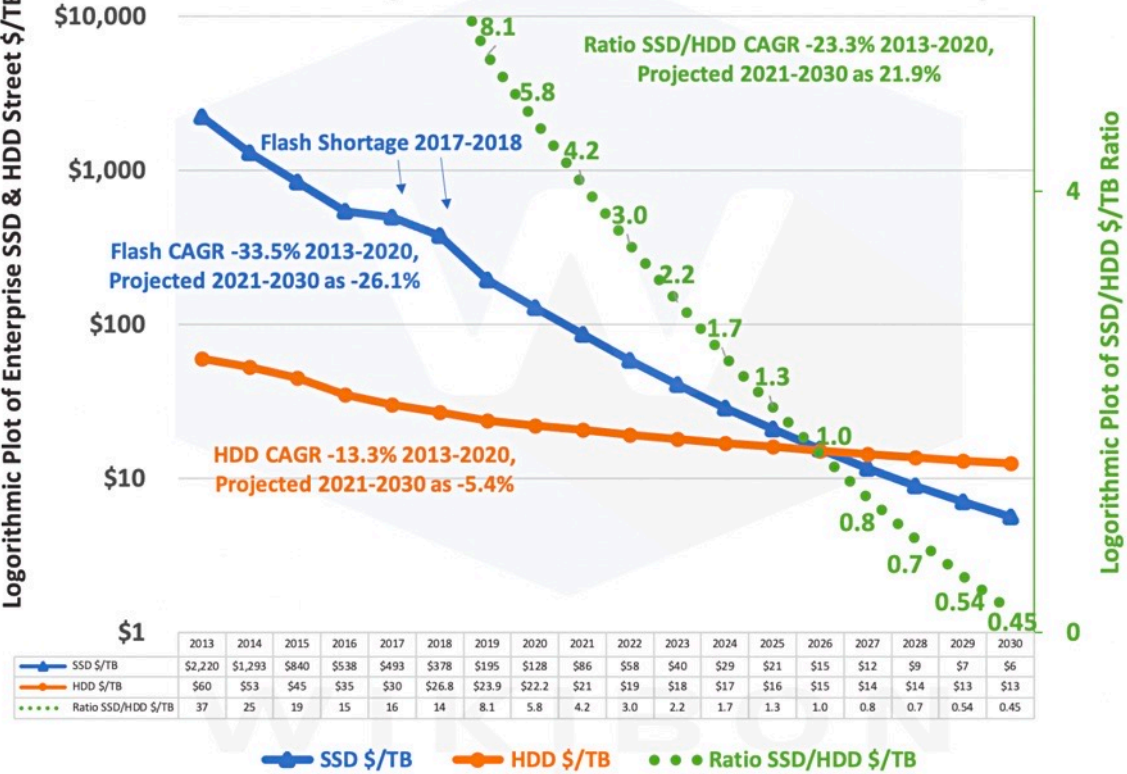


Source: © Wikibon, 2021

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

Source: © Wikibon, 2021.

SSD is Cheaper than HDD in 2026 and Beyond

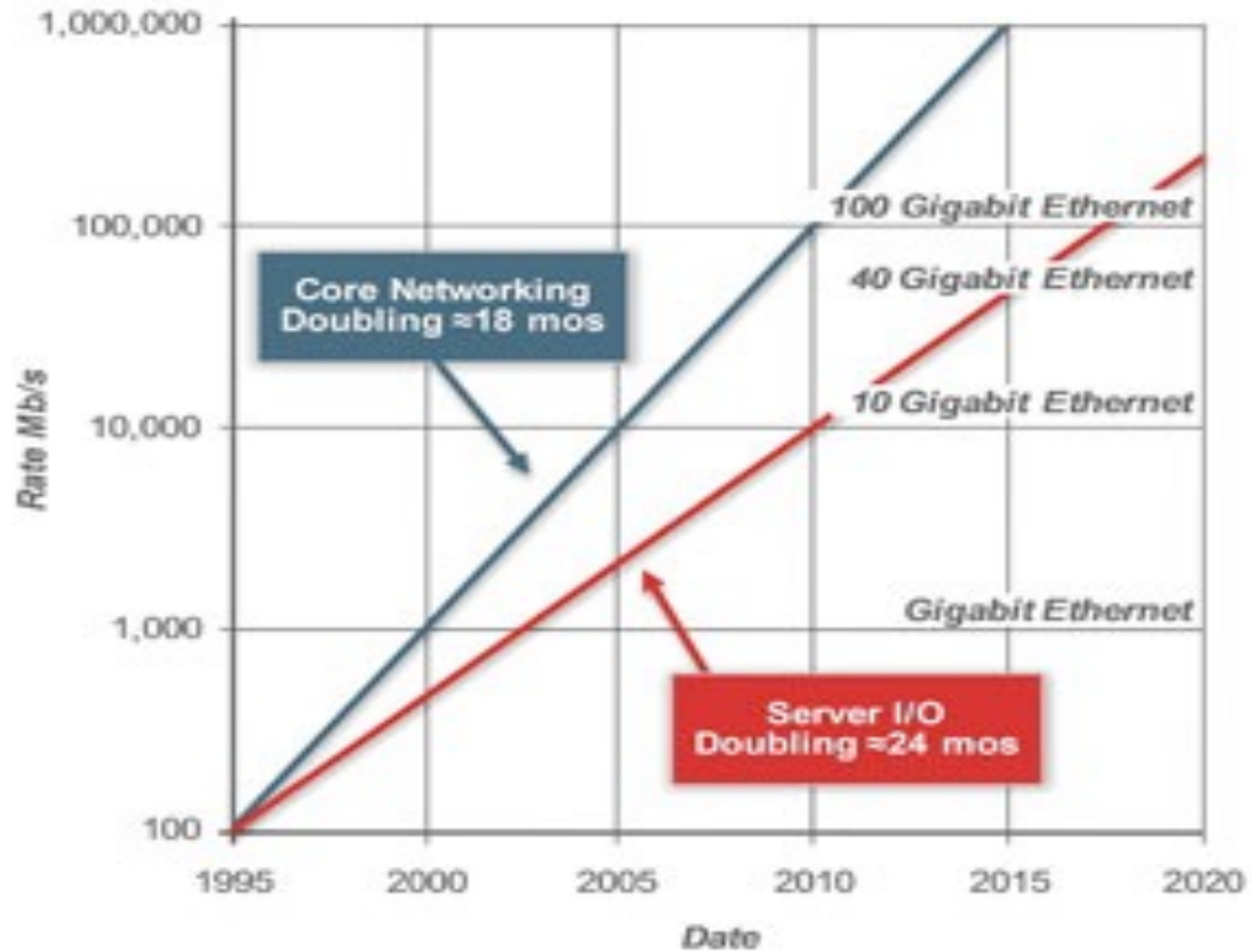


Source: © Wikibon, 2021. The 2021-2030 Enterprise Street Prices are Projected by Wikibon using Wright's Law

Figure 4 - SSD/HDD Pricing Ratio 2013 - 2030

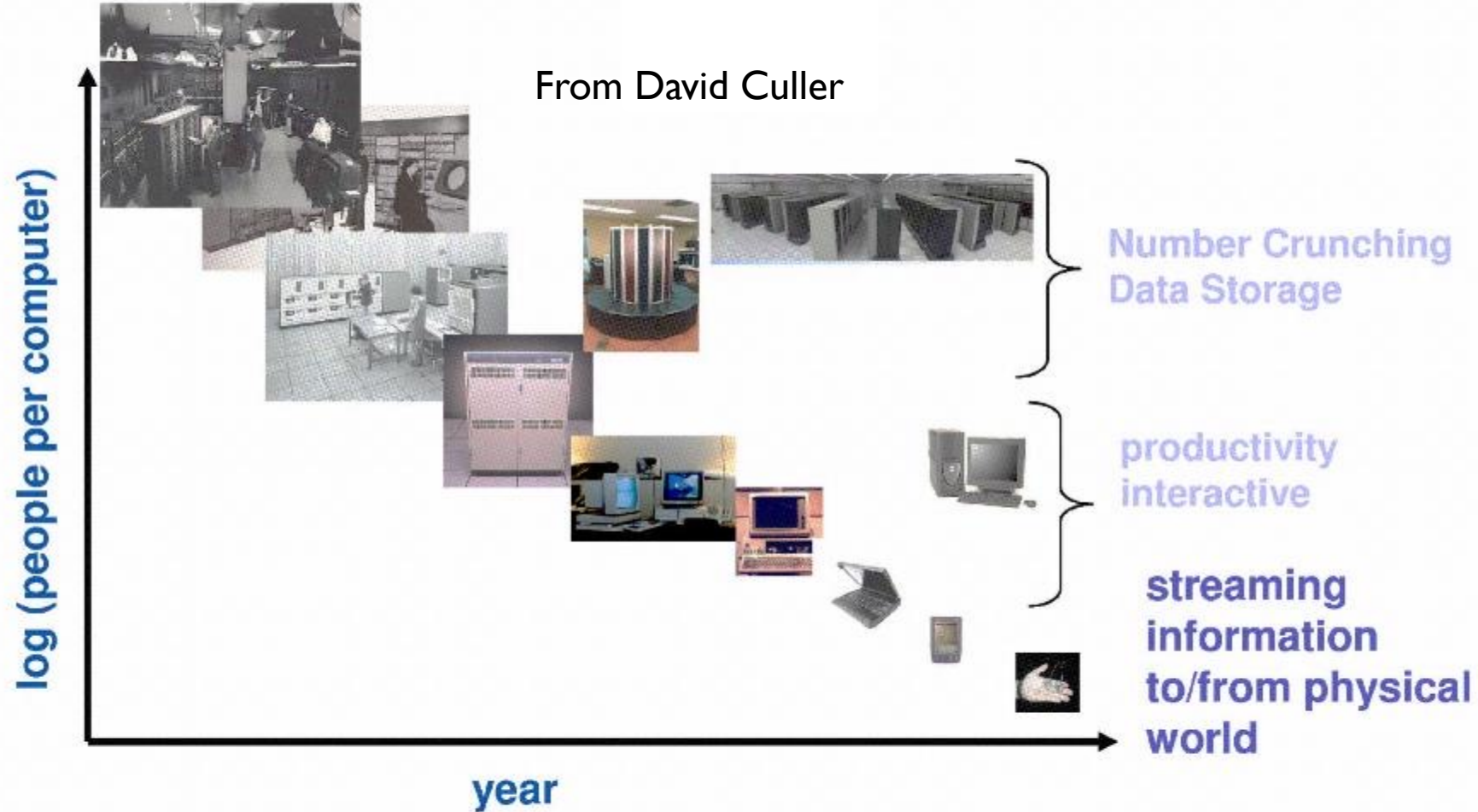
Source: © Wikibon, 2021.

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

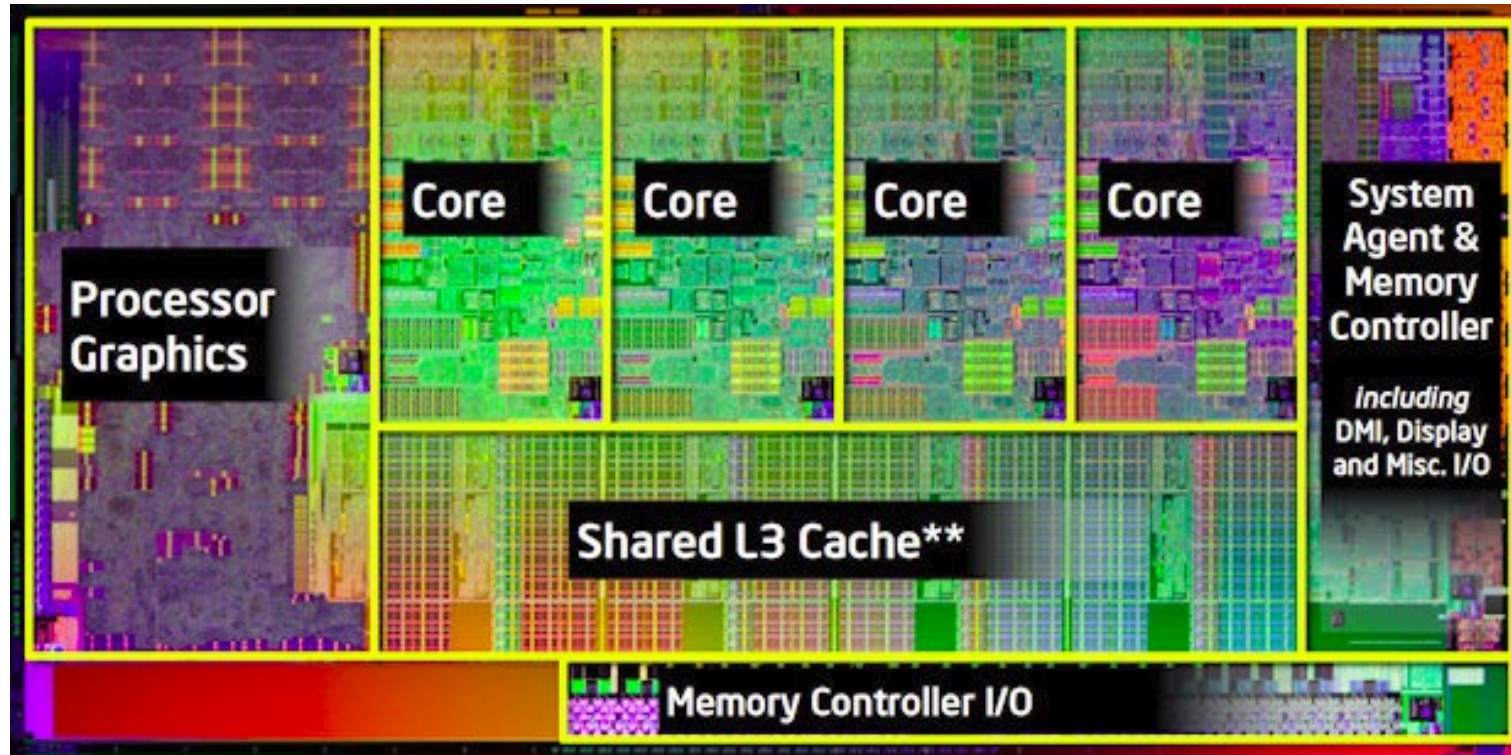
Jeff Dean: “Numbers
Everyone Should Know”

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!

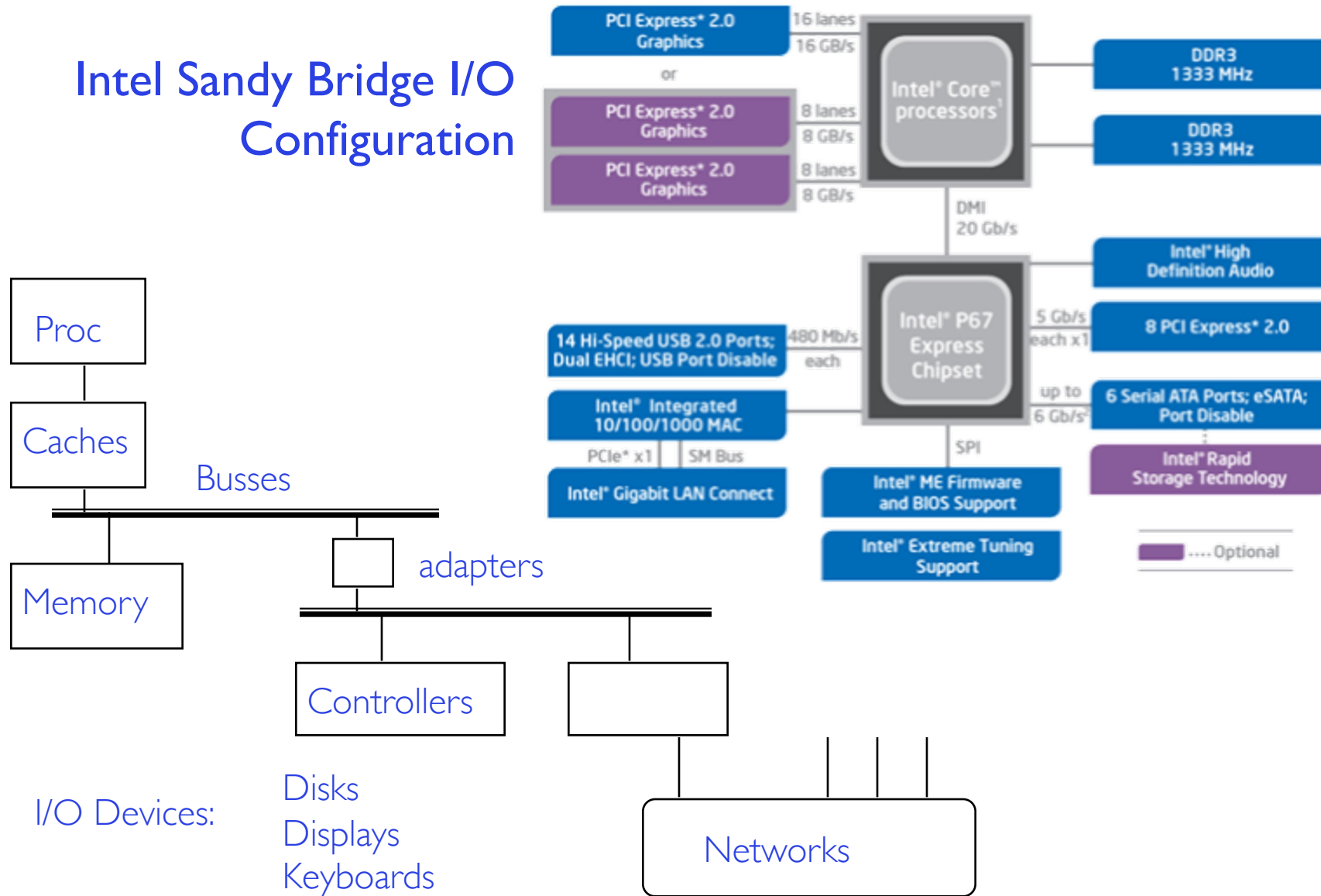
A Modern Processor: Intel Sandy Bridge



- 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

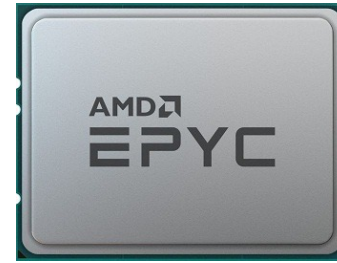
HW Functionality comes with great complexity!

Intel Sandy Bridge I/O Configuration

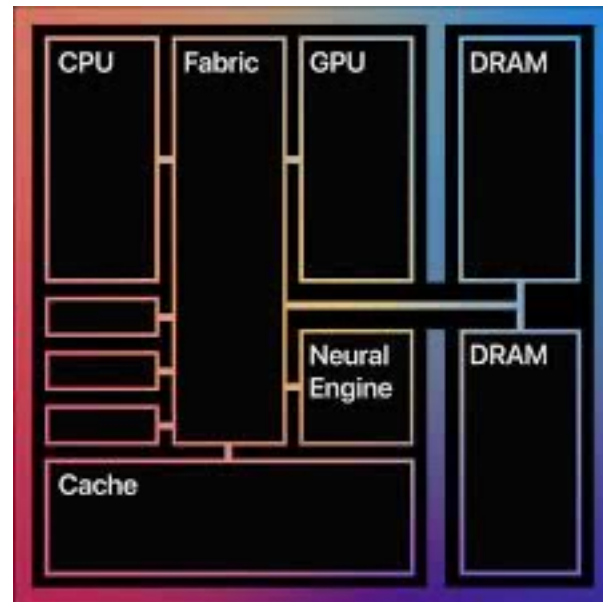


Not only specialized processors...

- Multi-core processors
 - Intel Xeon Phi: 64+ cores
 - AMD Epyc: 64+ cores



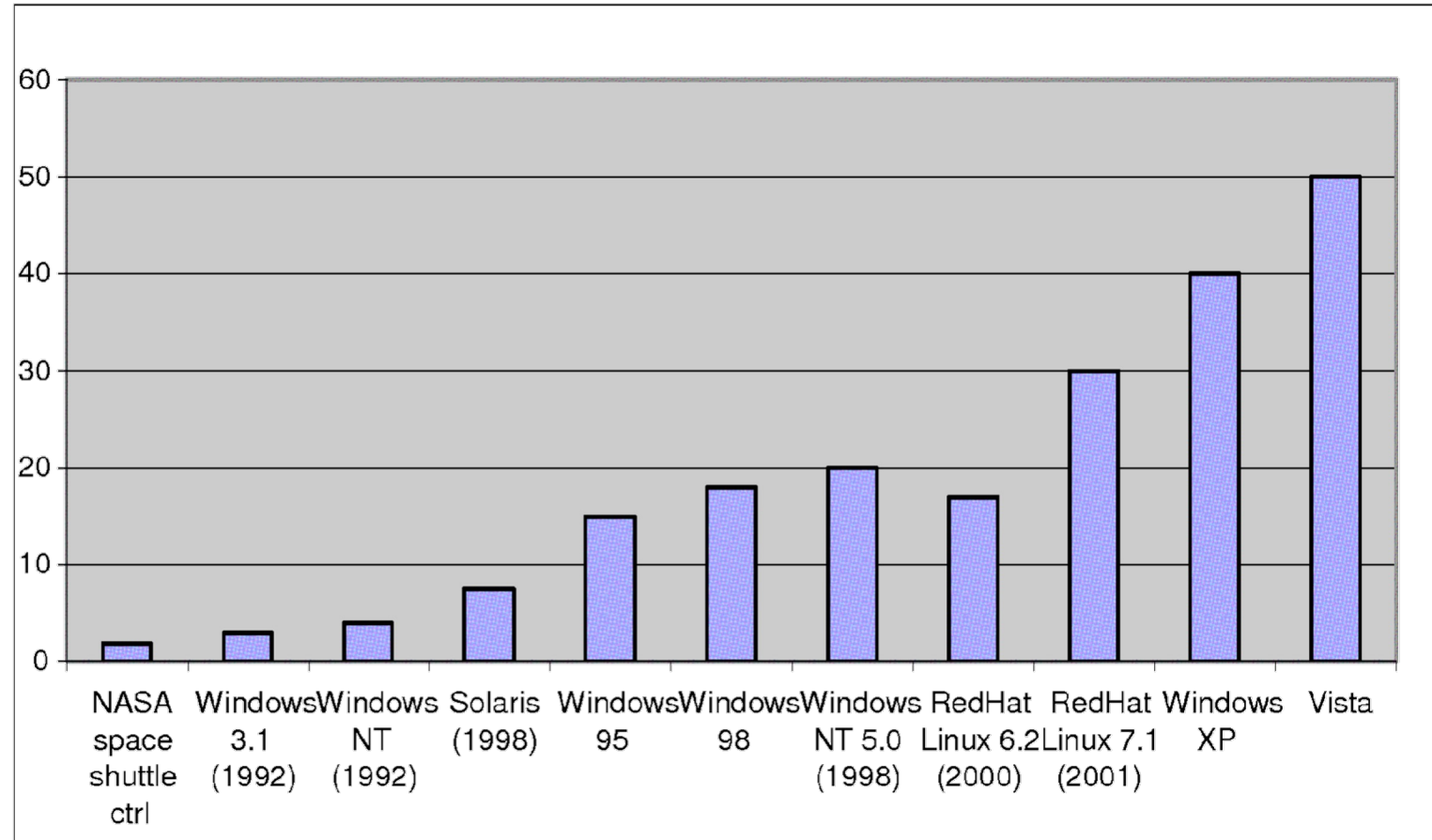
- System On a Chip (SOC): accelerating trend

An infographic detailing the specifications of the Apple M1 chip. It features a central M1 logo surrounded by various performance metrics and features:

- 5 nanometer process
- Machine learning accelerators
- 16-core Neural Engine (11 trillion operations per second)
- Thunderbolt / USB 4 controller
- Media encode and decode engines
- 16 billion transistors
- Up to 8-core GPU
- 8-core CPU
- Advanced image signal processor
- Secure Enclave
- Unified memory architecture
- Industry-leading performance per watt

Increasing Software Complexity

Millions of lines of
source code

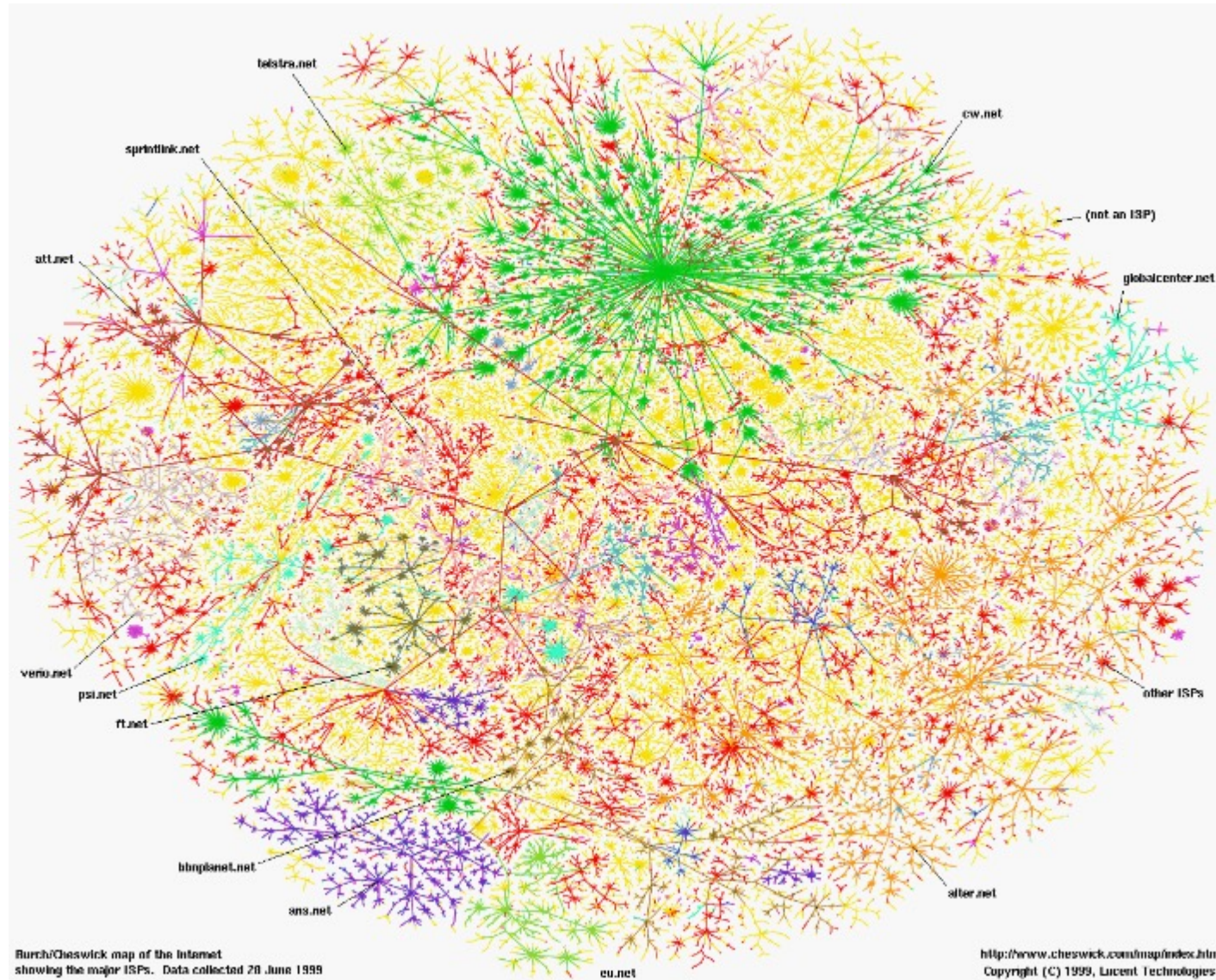


From MIT's 6.033 course

Everything going distributed

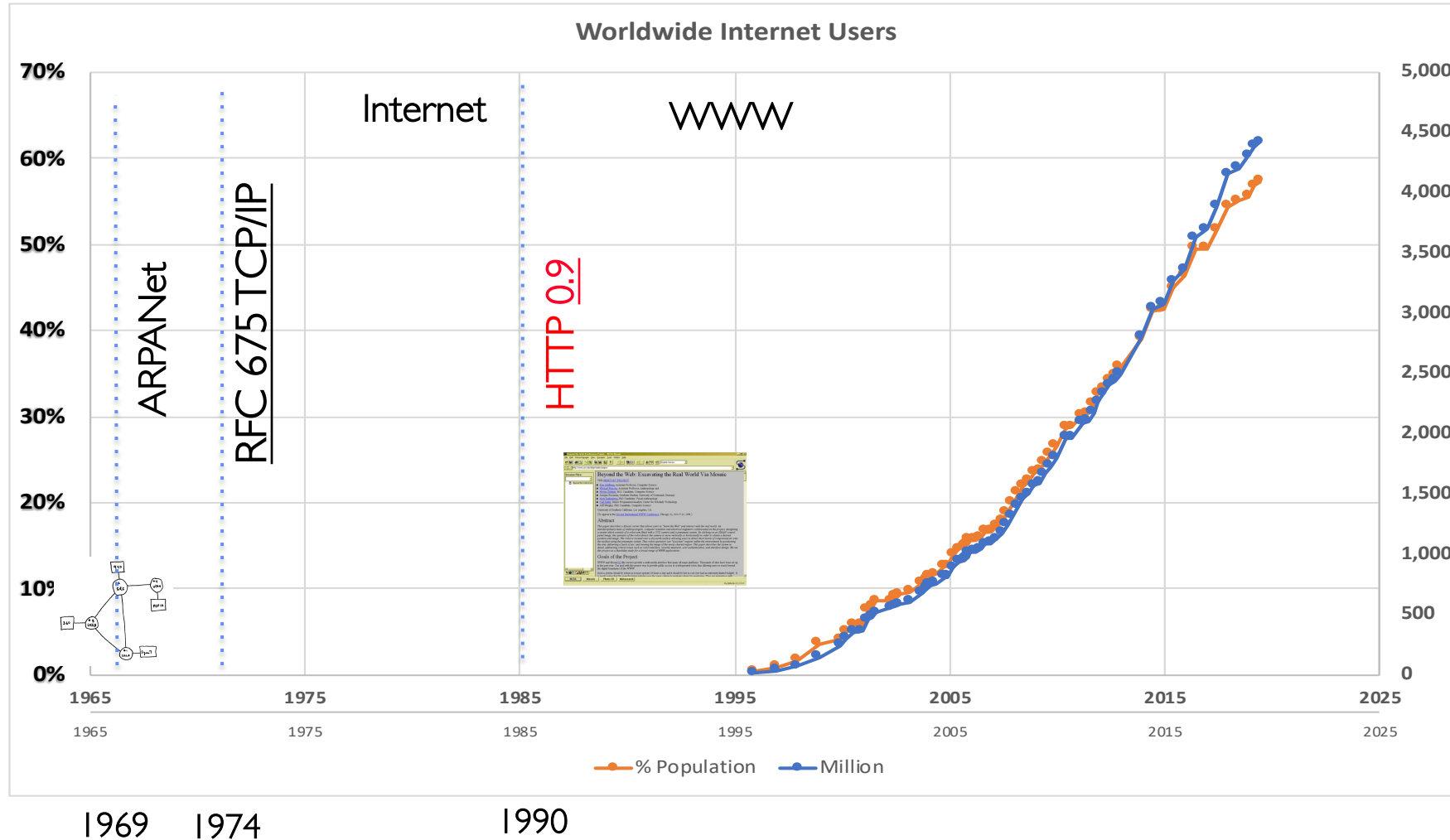
- Need to scale
 - Machine learning workloads, especially large language models
 - Big data analytics
 - Scientific computing
 - ...
- Everything is connected!

Greatest Artifact of Human Civilization...



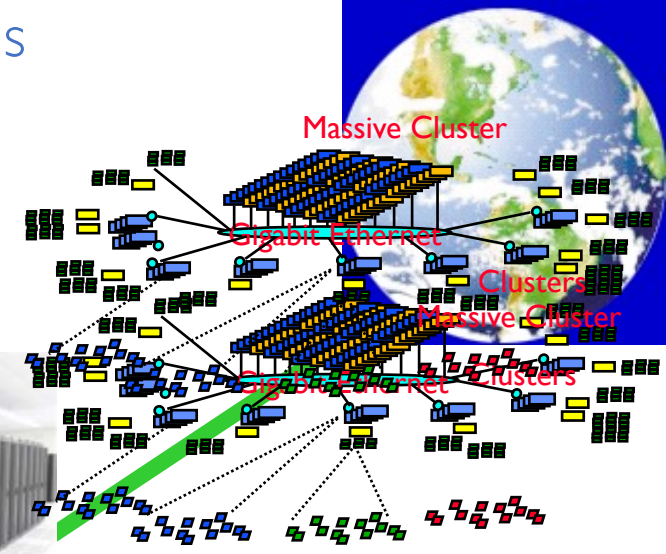
Map of the Internet

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



Internet
Connectivity

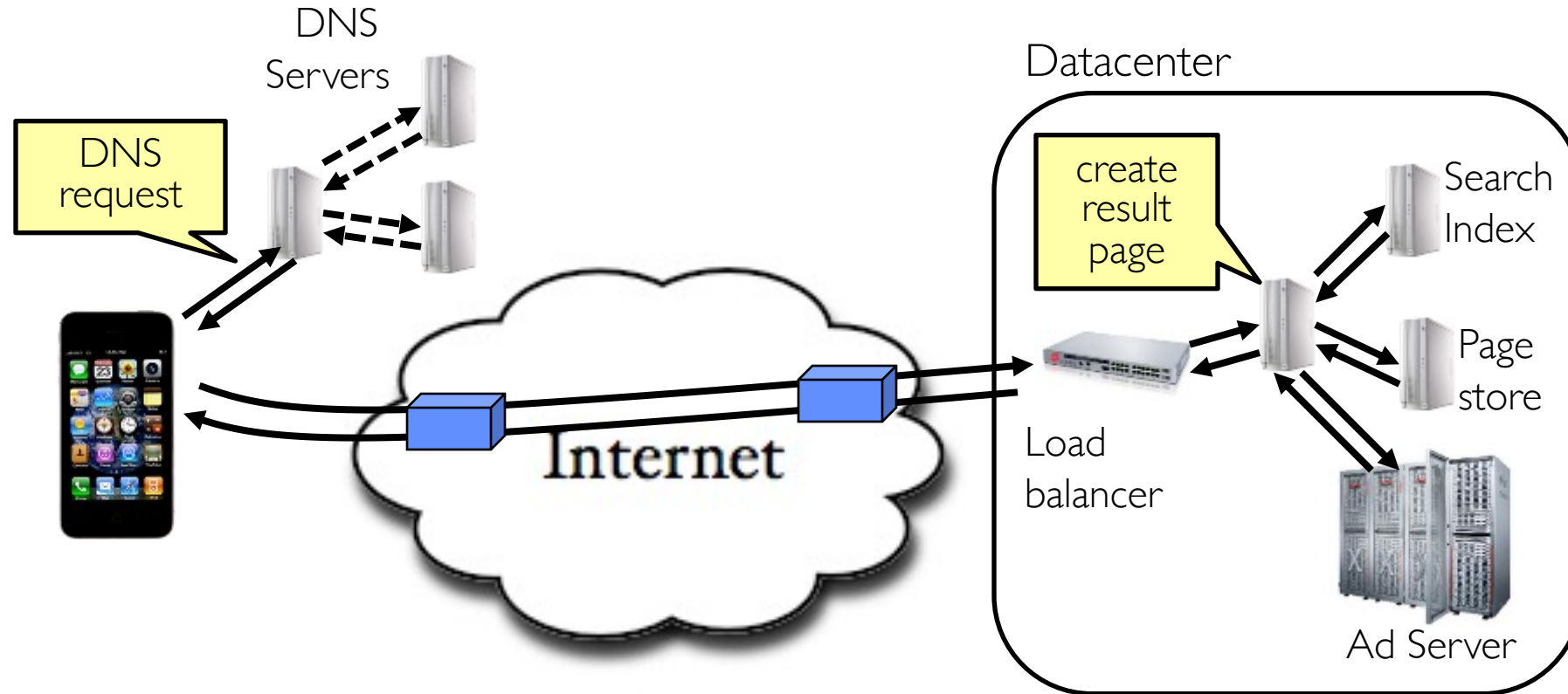
Scalable, Reliable,
Secure Services



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 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: February 29 (next Thursday)

Discussion

- What is an operating system?
- Operating systems vs. computer systems vs. system software
- Operating systems vs. distributed systems vs. networked systems vs. storage systems vs. database systems

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/PZaRCka.aspx>