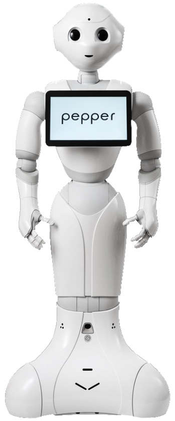
CS302

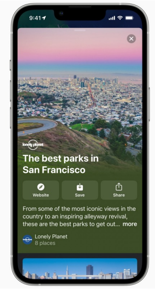
Introduction to Parallelism & Concurrency

Spring 2025

Arkaprava Basu & Babak Falsafi

parsa.epfl.ch/course-info/cs302





Who's Who

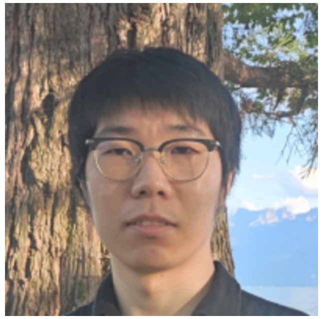
- Prof. Babak Falsafi
  - Computer Architecture
  - Post-Moore Servers, Sustainable Clouds
- Prof. Arka Basu
  - Operating Systems & Computer Architecture
  - GPU systems, virtual memory, security
- TAs:
  - Ayan: cloud-native servers
  - Pooria: rack-scale computing
  - Yuanlong: virtual memory

Arka

Babak

Ayan

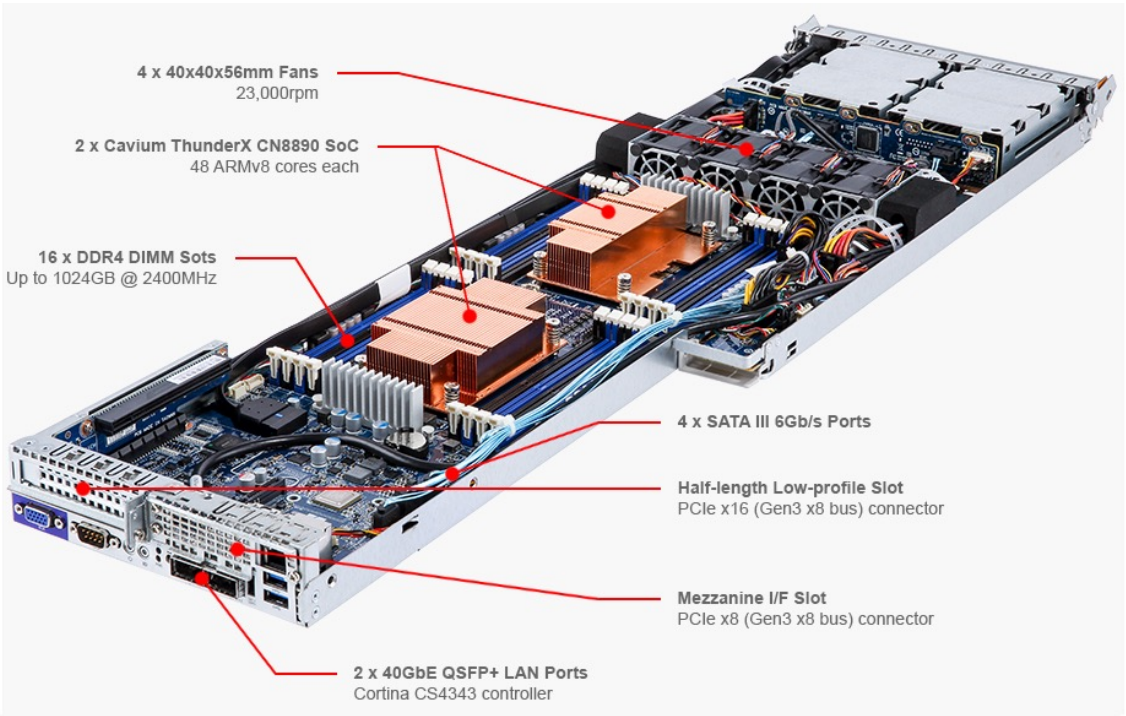
Pooria


Yuanlong

Who's Who

- SAs:
  - Patrick Pataky
  - Rayan Boucheny
  - Simon Desmaison
  - William Robert

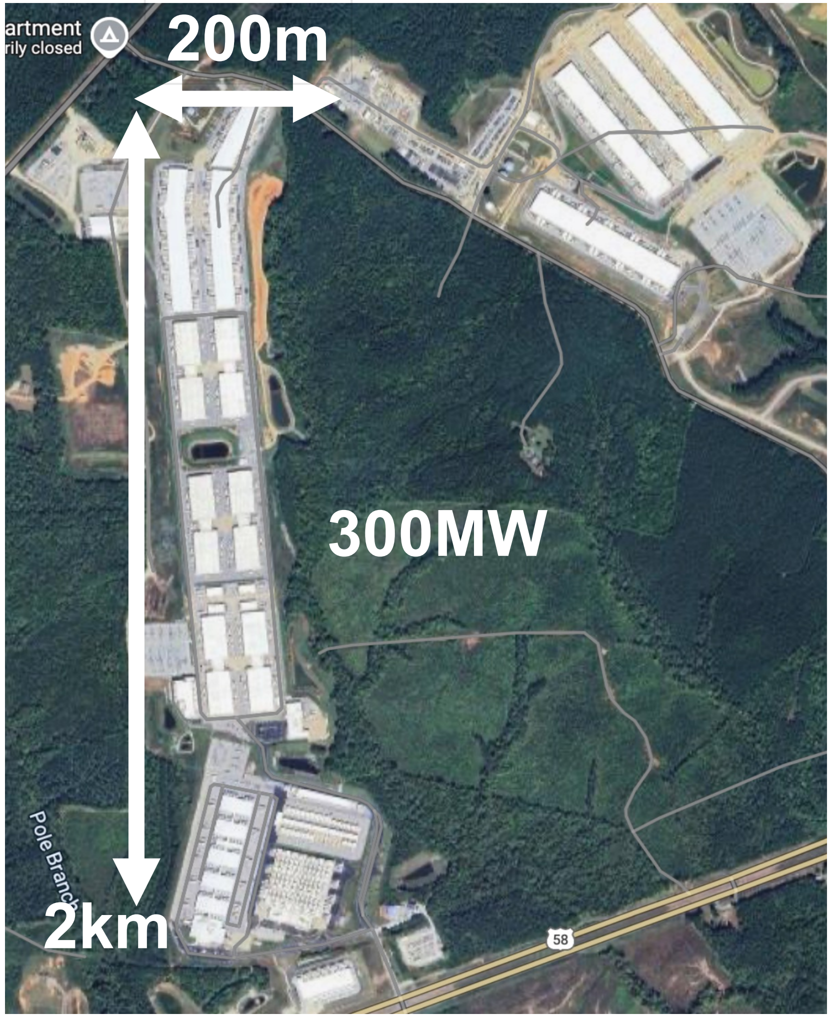
Babak’s research: Post-Moore Servers


4 x 40x40x56mm Fans
23,000rpm
2 x Cavium ThunderX CN8890 SoC
48 ARMv8 cores each
16 x DDR4 DIMM Sots
Up to 1024GB @ 2400MHz
4 x SATA III 6Gb/s Ports
Half-length Low-profile Slot
PCIe x16 (Gen3 x8 bus) connector
Mezzanine I/F Slot
PCIe x8 (Gen3 x8 bus) connector
2 x 40GbE QSFP+ LAN Ports
Cortina CS4343 controller

- Design servers with cell phone energy efficiency
- Thunder X: first industrial-grade ARM-based server CPU

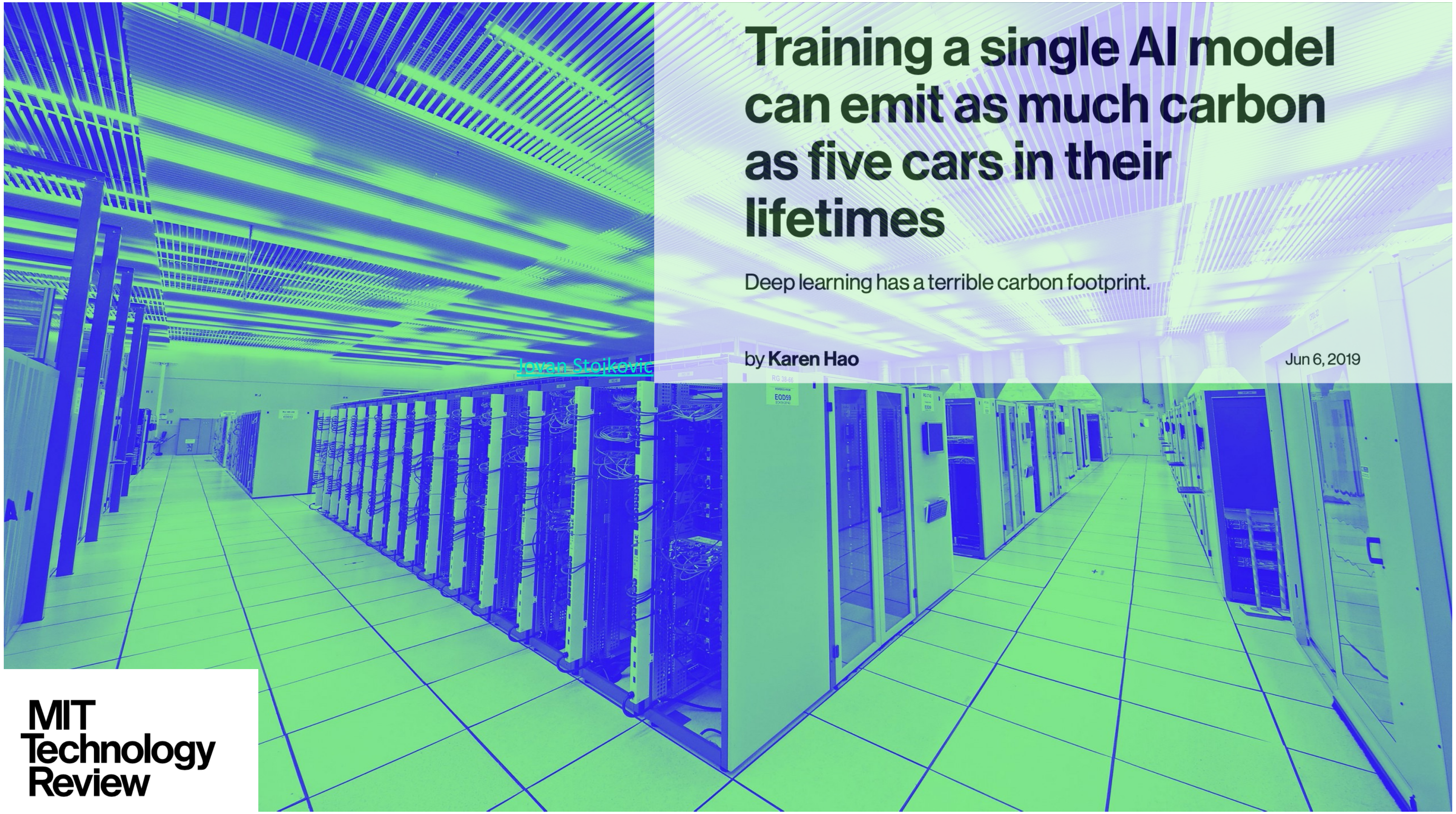
Babak’s research: Datacenters

- A million home-brewed servers
- Centralized to exploit economies of scale
- Network fabric w/ μ-second connectivity
- At physical limits
- Need sources for
  - Electricity
  - Network
  - Cooling



Datacenter, Boydton VA

Babak’s research: Sustainable Datacenters



Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by Karen Hao

Jun 6, 2019

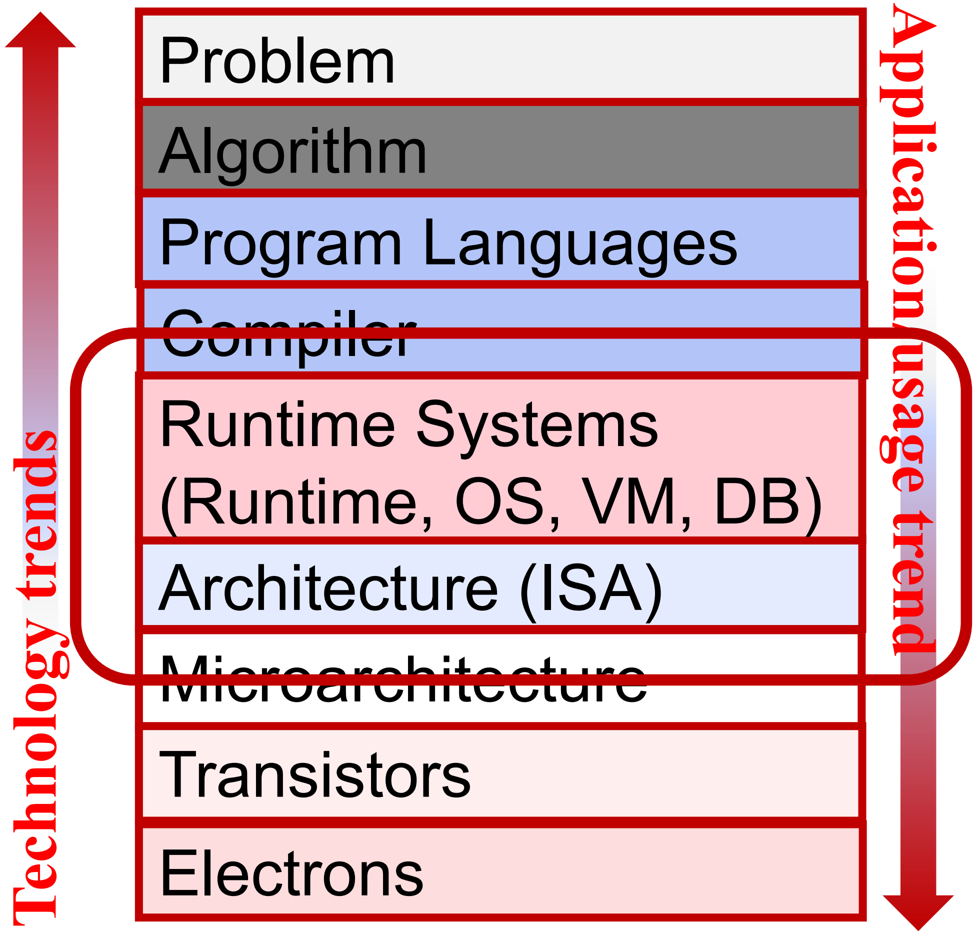
Jovan Stojkovic

MIT Technology Review

Arka’s research: Explorations across the compute stack

Computer Systems Lab
https://csl.csa.iisc.ac.in/

Virtual memory,
Memory management,
HW-SW co-design



The compute stack

Recent research focuses on the software and the hardware for GPUs

Schedule for this semester

M	T	W	T	F
	18-Feb	19-Feb	20-Feb	21-Feb
24-Feb	25-Feb	26-Feb	27-Feb	28-Feb
3-Mar	4-Mar	5-Mar	6-Mar	7-Mar
10-Mar	11-Mar	12-Mar	13-Mar	14-Mar
17-Mar	18-Mar	19-Mar	20-Mar	21-Mar
24-Mar	25-Mar	26-Mar	27-Mar	28-Mar
31-Mar	1-Apr	2-Apr	3-Apr	4-Apr
7-Apr	8-Apr	9-Apr	10-Apr	11-Apr
14-Apr	15-Apr	16-Apr	17-Apr	18-Apr
21-Apr	22-Apr	23-Apr	24-Apr	25-Apr
28-Apr	29-Apr	30-Apr	1-May	2-May
5-May	6-May	7-May	8-May	9-May
12-May	13-May	14-May	15-May	16-May
19-May	20-May	21-May	22-May	23-May
26-May	27-May	28-May	29-May	30-May

- Class intro
  - Logistics
  - Grades
  - Overview of topics
- Thursday
  - Lecture: Metrics for computing
  - Exercise session: Intro and examples of Amdahl’s law
- Lab session on Friday
  - No lab for this week

Roadmap

- Class intro
- Parallel computing
- Shared memory
- Message passing
- Memory consistency
- Synchronization
- Continuations/RPC
- Multithreading
- GPUs
- Beyond GPUs

Where do I find information about CS302?

Info page:

parsa.epfl.ch/course-info/cs302

- Schedule
- Contact information

Moodle page (search for CS302):

https://moodle.epfl.ch/course/view.php?id=18349

- Announcements, discussion boards
- Handouts & supplementary reading material
- Homework & programming assignments
- Grades & solutions

Where can I discuss/ask questions about CS302?

Edstem page:

https://edstem.org/eu/courses/2076/discussion

- Ask private and public questions to TAs and SAs
- Public sharing of code snippets not allowed
- High-level discussions about concepts taught in class allowed
- Discussion of solutions of homeworks and assignments NOT allowed

Important note:

- All official announcements will be through Moodle
- Edstem only for discussions between students and TAs

Logistics for the Course

Class times:

- Lectures: **Tue 10:15am-12:00pm, CM13**
  Thu 10:15am-11:00am, CM 4
  - Recordings will be provided on Moodle

- Exercise: **Thu 11:15am-12:00pm, CM 4**

- Lab: **Fri 10:15am-12:00pm, CM 011 & CM 012**

Academic Honor Code

- Do not forget the EPFL Honor Code

- Solutions/Code submitted must be **fully** your own work

Examples of academically dishonest behavior:

- Copying text or code from others or from solutions of previous years
- Working on assignments with other students that are not part of your group

- **We will be checking!** Be safe, be honest.

What is the point of this class?

- Our computing world is parallel
- What are the underlying computing technologies in this world?
- How do we construct well-behaved parallel/concurrent software?

We divide this design task into:

1. Types of Multiprocessors
   Multicore/Manycore, Vector/SIMD, GPU, Clusters

2. Parallel/concurrent programming
   Languages & runtimes that express parallel/concurrent execution:
   OpenMP, MPI, coroutines, and CUDA, for CPU & GPU

What knowledge does CS302 assume?

- High-level languages and data structures
- Introduction to computer architecture
  - Scalar pipeline execution
  - Dynamic scheduling and Out of Order
  - Caches
- Basic C/C++ programming

Programming Assignments

Demonstrate your ability to program multiprocessors

- Use concepts taught in lectures to write fast programs
- One CPU assignment on OpenMP
- One CPU assignment on MPI
- One GPU assignment using CUDA
- Assignments have multiple parts
- 25% of total grade

Programming Assignments

- Can work on assignments in the lab sessions
  - TAs and SAs available to answer questions
- Assignments released every ~4 weeks
- Done in pairs of two students
- All code is auto-graded, and must comply with the format!

Assignment Platforms

- Lab sessions are in person every Friday
- First lab session NEXT FRIDAY

February 28th, 10:15am-12:00pm, CM 011 & CM 012

  - Walk through readings, work on programming assignments and homeworks

- **You have to use your laptop!**
  - PCs are not available in CM 011 & CM 012
  - OpenMP support is baked into gcc for Unix-like OS
- For access to GPUs, we use *SCITAS @ EPFL*
  - Compile and execute your code remotely
  - Access will be granted to you if you remain enrolled in this course

Homework

- Total eight homeworks

- Assigned after the lecture
  - First homework (not-graded) is posted on Moodle

- Covers concepts presented in class

- 20% of total grade
  - Grading sampled every week (not all questions graded)

Exercise Sessions

- Exercise sessions are in person every Thursday after the lecture
- First exercise session THIS THURSDAY

- Purpose of the exercise sessions:
  - Example demo programs and tutorials
  - Solve example questions on topics covered in lectures
  - Work on homework and assignments

Grading/Regrading

- Grades (curved)
  - Assignments 30%
  - Homework 20%
  - Midterm 20%
  - Final 30%

- Regrading for up to a week after the grade release
  - Please contact TAs to ask for a regrade

More info

- Lecture slides posted on Mondays (or before)

- Grades will be posted on Moodle
  - Within two weeks of the submission deadline
  - With high-level feedback

- TAs will answer queries during exercise and lab sessions

Readings

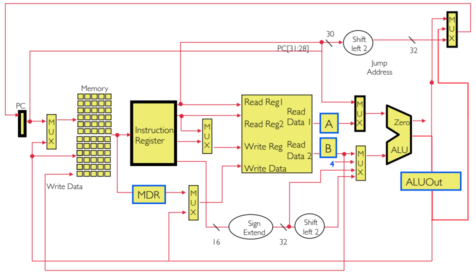
- Meant to supplement lecture material
- This week:
  - Parallel Computer Architecture: A Hardware/Software Approach, D. Culler & E. Singh, Chapter 1
  - Will post PDFs on Moodle
- Next week:
  - OpenMP programming and primitives
- The readings for *Consistency* are much longer
  - Spread them out
  - Carefully think about content while you read

Overview of topics

Where do we stand?

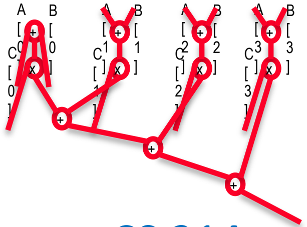
Background:

Architecture



CS 200

Software Construction



CS 214

Systems

CS 202

Related:

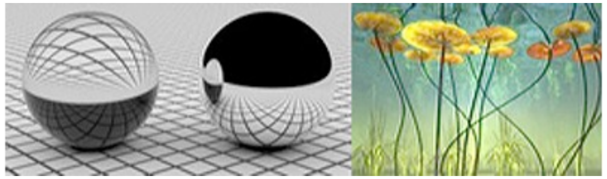
Parallel Computing

CS 302

Databases

CS 300

Intro to OS



CS 323

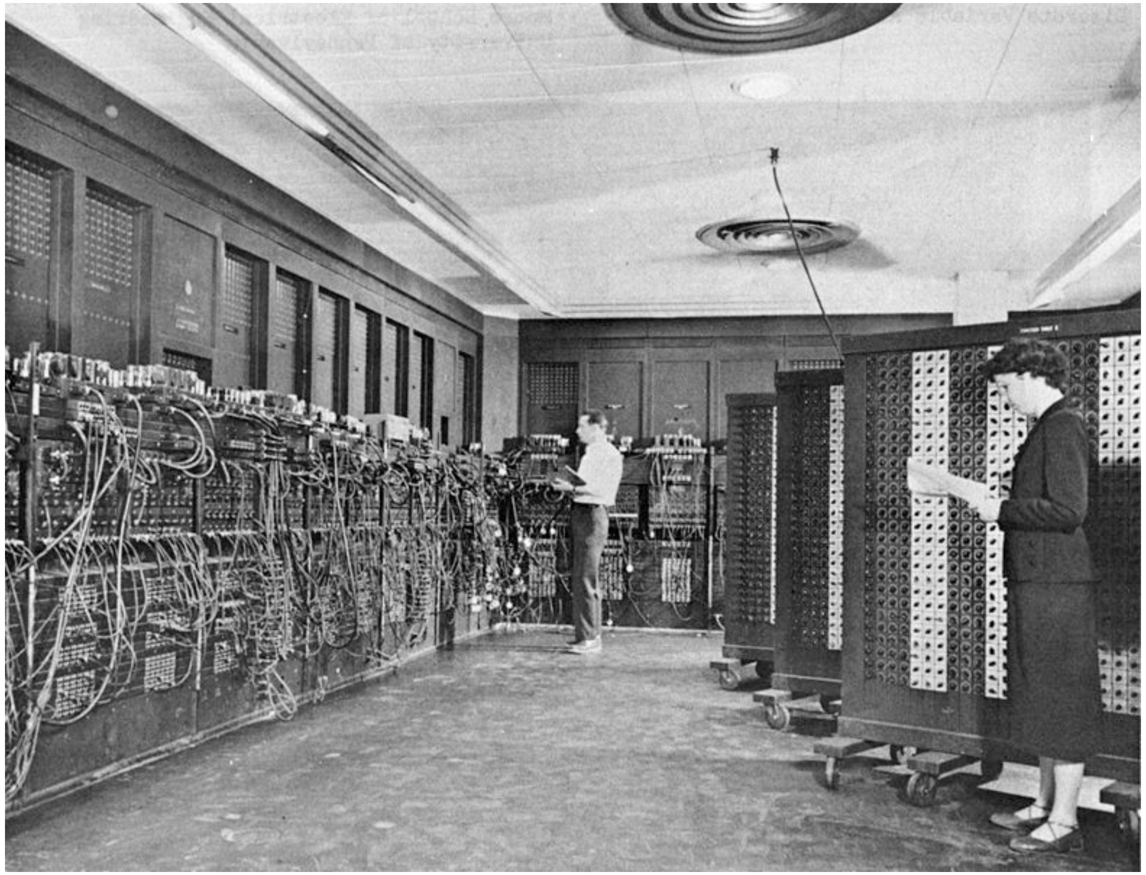
Where do we stand?

- Computer architecture
  - Instruction Set Architecture (ISA)
  - Data/Control Path
  - Memory and caches
- Software construction
  - Concurrent data structures
- Computer systems
  - Virtual memory
  - Processes and threads
- Parallelism & concurrency (this class)
  - Shared memory
  - Message passing
  - GPUs

Where did it all start? ENIAC

- University of Pennsylvania
- Eckert and Mauchley
- Cost $486,804.22, in 1946
- 5000 ops/second
- 19K vacuum tubes
- Power = 200K Watts

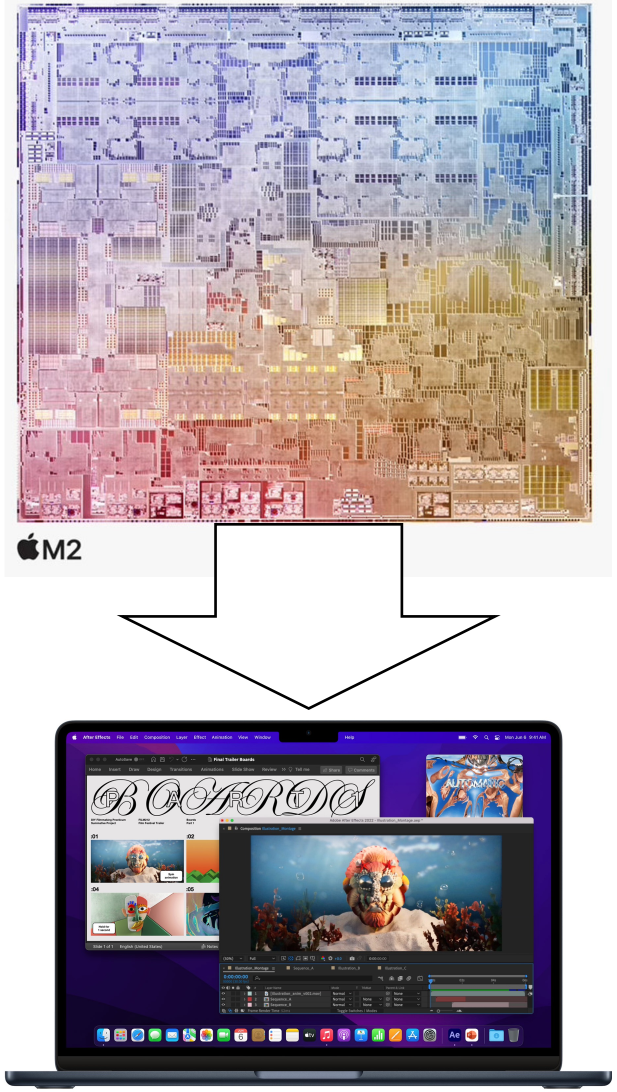
[picture from Wikipedia]



67 m^3

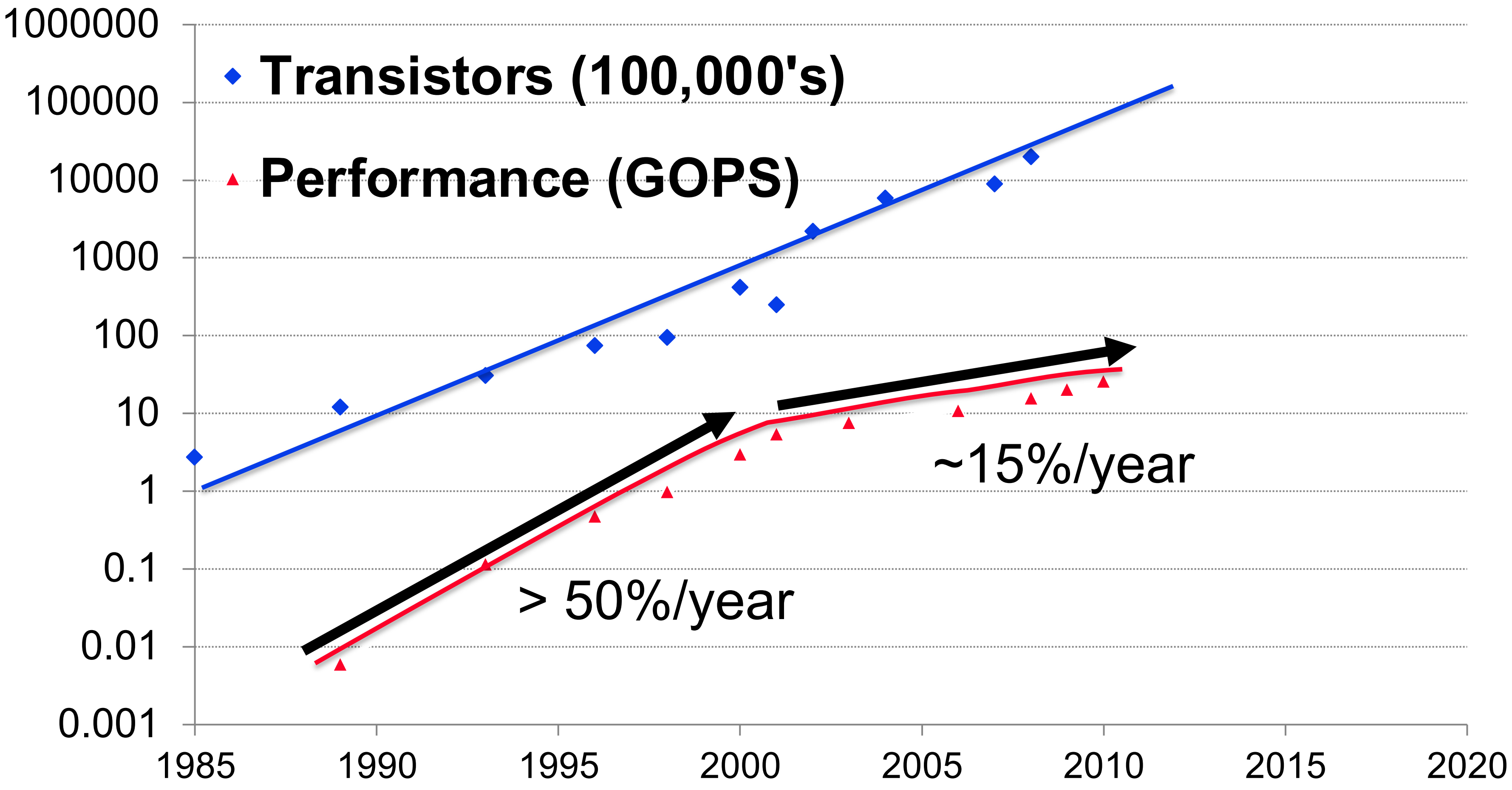
Where are we today? Apple M2 Chip

- 20 billion transistors
- 8 CPU, 10 GPU, 16 Neural cores
- 16 MB on-chip memory (SRAM)
- 24 GB off-chip memory (DRAM)
- 3.49 GHz
- Roughly 20W



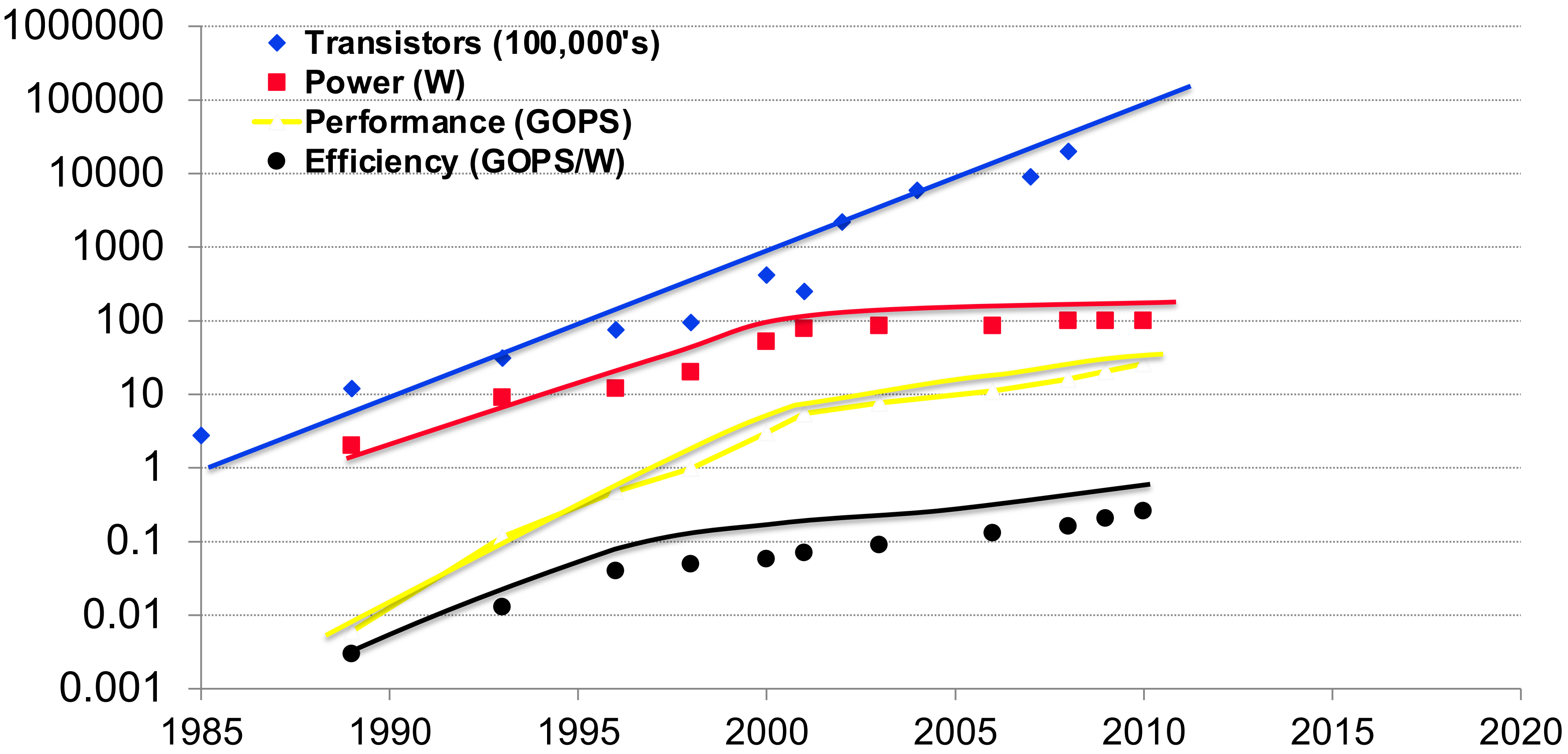
630 cm^3

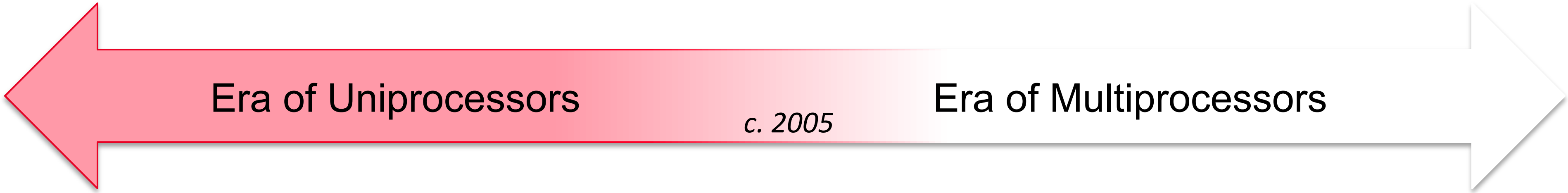
The chips got bigger but not faster!!!!



[source: Prof. Tom Wenisch, Michigan]

Why? Must keep power at ~100W!





What is Power?

Dynamic power $\propto V^2F$

V = Operating voltages

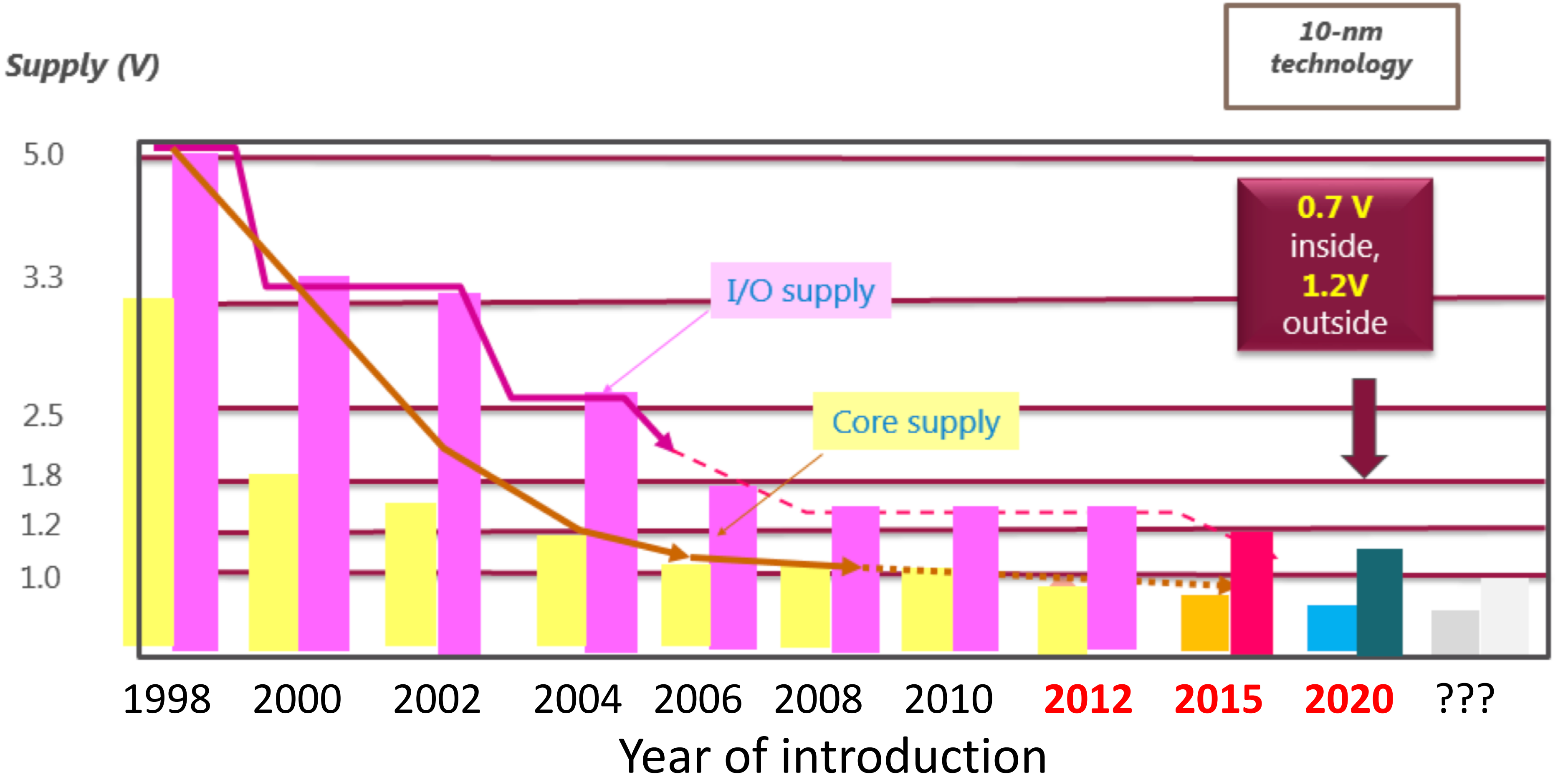
F = Clock frequency

CPU type	Power	Constraint
IoT/Embedded/Mobile	< a few W	Battery usage
Laptop	< 10s of W	Battery + heat
Desktop/Server	~ 100 W	Cooling
Supercomputer	~ 300 W	Cooling + electricity

What happened to Power?

- Voltages used to go down (Dennard’s Law)
  - From 5v (1970’s) to 1v (2000’s)
  - Power $\propto V^2F$
  - Power went down
  - But, voltage is squared!
  - Gave us enough room to increase clock frequency

(Supply) Voltages have leveled!



[source: Etienne Sicard. Introducing 10-nm FinFET technology in Microwind. 2017. hal-01551695]

For four decades 2x transistors every 2 years

Moore’s Law:

- More trans, faster CPU’s
- Clocks from 1 MHz to 2GHz
- Parallel microarchitecture: superscalar + pipelining
- Perf: 2.5x per 2 years

Lowered chip power with lower voltages

Ran sequential code, one thread/program

CPU in 1980

CPU in 1990

CPU in 2000

Voltages stopped going down

With more transistors:

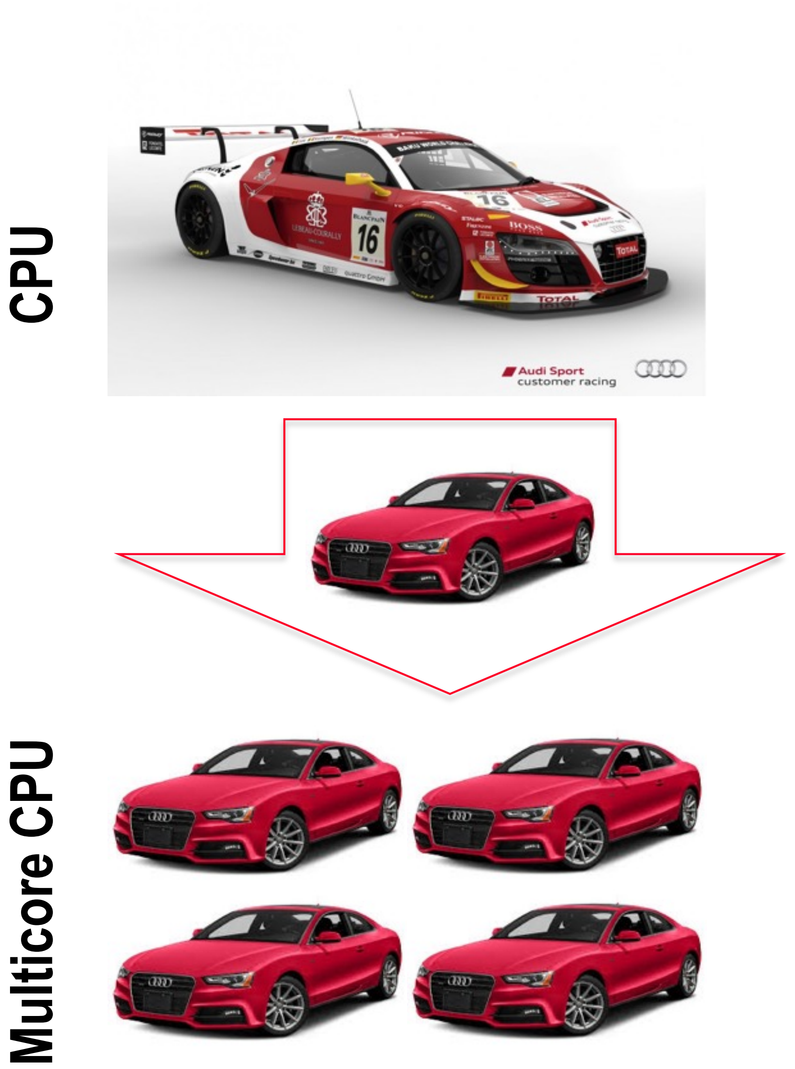
- Scale back core complexity
- Use cores of yesteryear
- Each core ➔ fewer joules/op

Analogy:

- Not quite a race car
- Handles nearly all cases

But, now……

- Need parallel software!



Exercise 1: Moore’s Law (of Performance)

CPU performance doubled in performance roughly every 18 months for four decades (58%/year).

Assume that Moore’s Law continues with doubling cores every two years, keeping the same cores and clocks and perfect parallel software, can multicores achieve the same increase in performance as before?

Answer: No (at best 2x/24 months, 41%/year)

Exercise 2: Moore's Law

Silicon density has been growing less than 2x every two years. The latest chips indicate an increase in transistor counts of 15%/year.

Assume keeping the same cores and clocks and perfect parallel software, how many years will it be before our performance doubles?

Answer ~ 5 years ➔ $(1.15)^5$

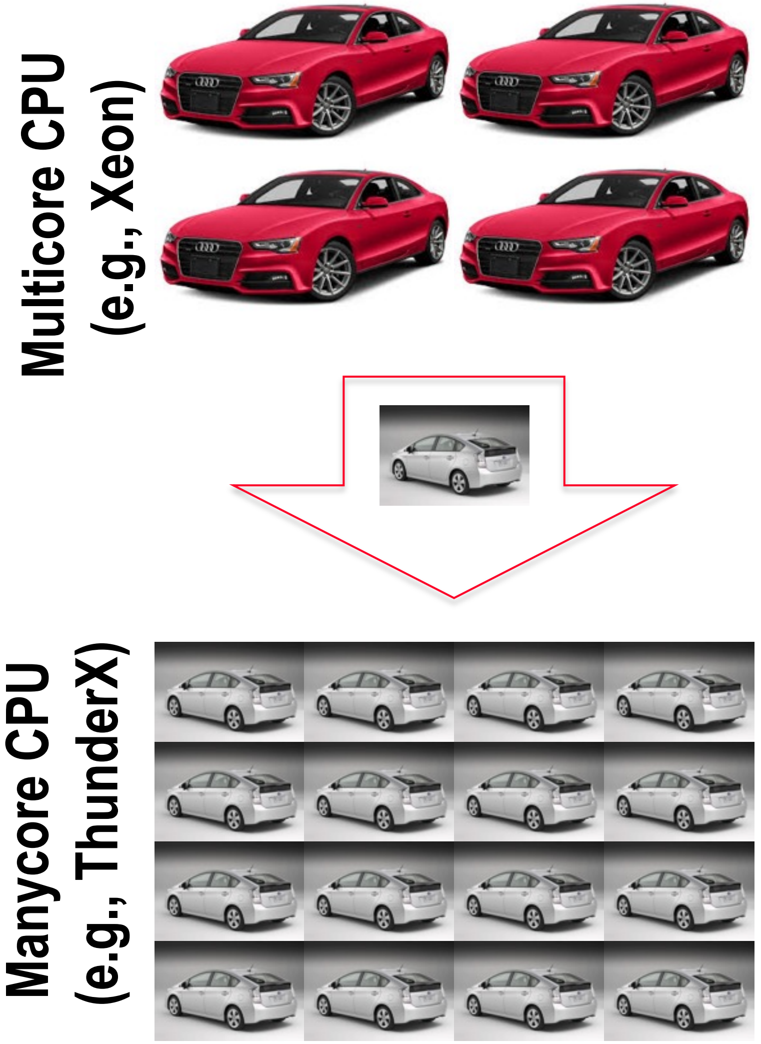
From Multicore to Manycore

Can no longer afford the general-purpose & high-perf

- Custom cores
- Reduced complexity
- Mobile efficiency

Analogy:

- Prius can handle city well
- But, limited at speed
- More work from parallelism



Exercise 3: Cores vs. Clock

Power $\propto V^2F$

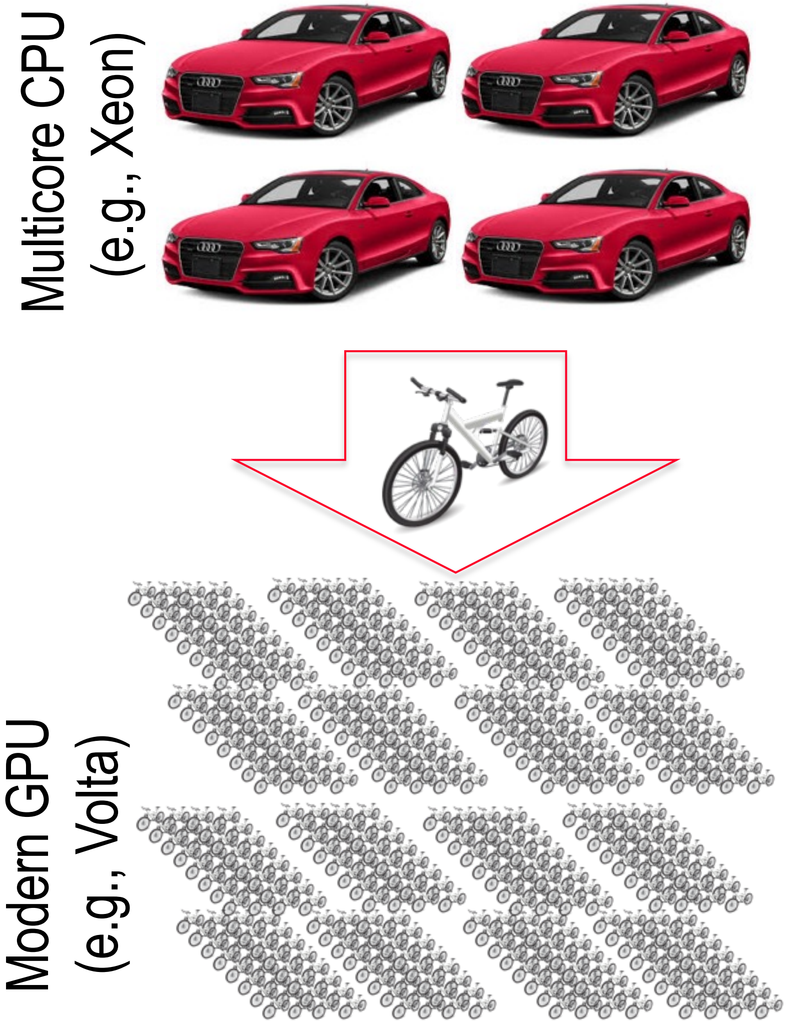
Assuming a 2GHz clock, voltages remain the same, to quadruple the number of cores (4x cores) while keeping the power the same, how much slower should the cores run at (slower clock rate)?

Answer: 500 MHz

GPUs are Massively Parallel Manycores

Further reduction in complexity:

- Five stage pipeline is minimum complexity
- GPU cores share microarchitecture in groups to save space
- Fetch one instruction, run on multiple data (SIMD)
- Program needs to run in groups



Custom Computing (not in this course)

Reconfigurable

- FPGAs, program in HDL
- Irregular parallelism
- Microsoft’s Catapult

Accelerator

- App-specific, min. work
- Program with DSL
- Google’s TPU

Connectivity

Single-chip up to multiple chips/single board

- Shared memory (read/write any address)

Across boards

- Distributed memory (read/write only local)
- Pass messages through a network

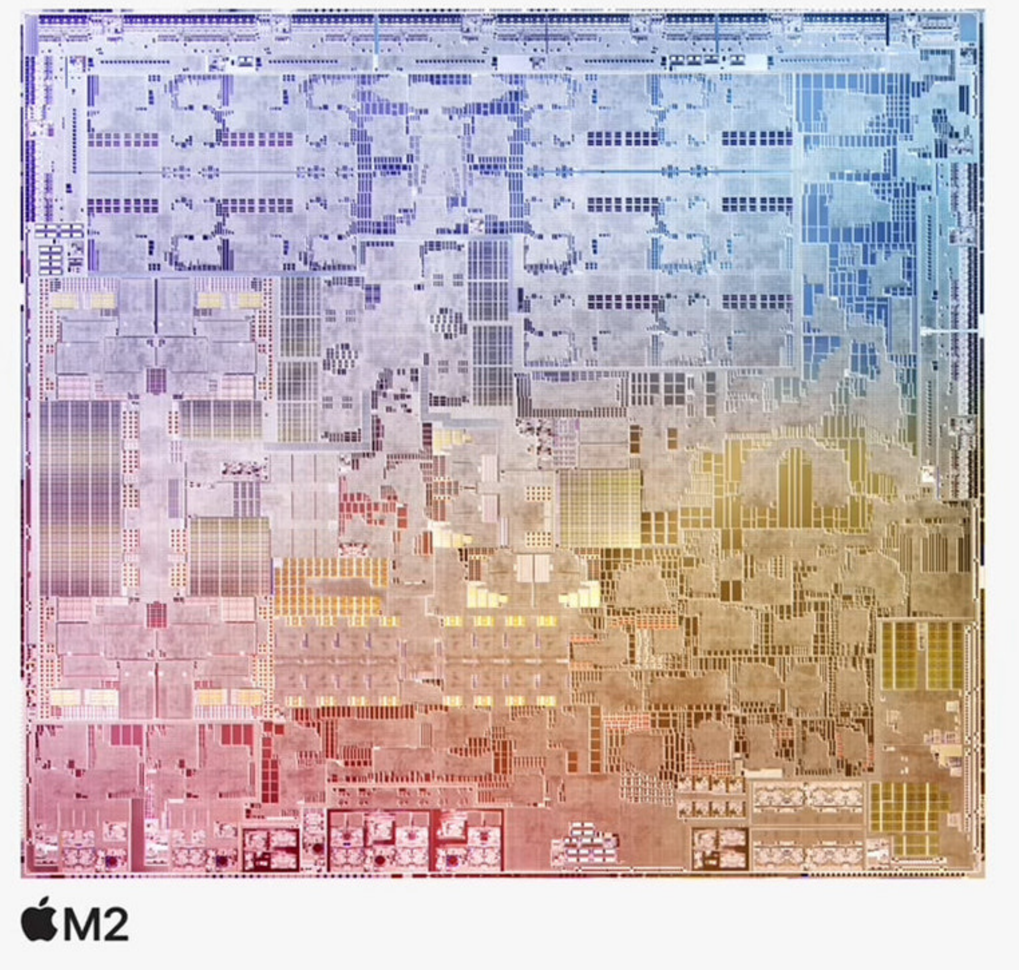
15-minute Break

Example: Smartphones & Tablets

- iPhone, Surface, Smartwatch
- Small devices used in everyday life
- Heterogeneous CPU/GPU devices
- Four low-power cores
  - 0.25 to 1 Watt per core
- Total power < 15 W
- Price < 1000 CHF

Mobiles use System-on-Chip (SoC)

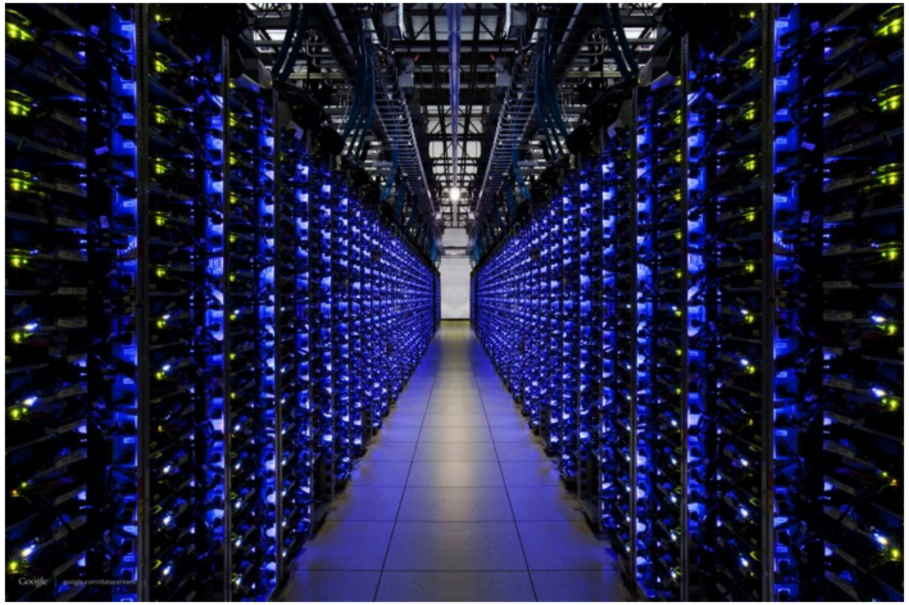
- 4 high-performance CPU cores
- 4 efficient CPU cores
- 10 GPU cores
- 16 neural engine cores
- Coherent shared memory
  - CPUs + Accelerators communicate on chip
  - Pointers equally valid on both



Apple M2 Chip

Example: Servers in Datacenters

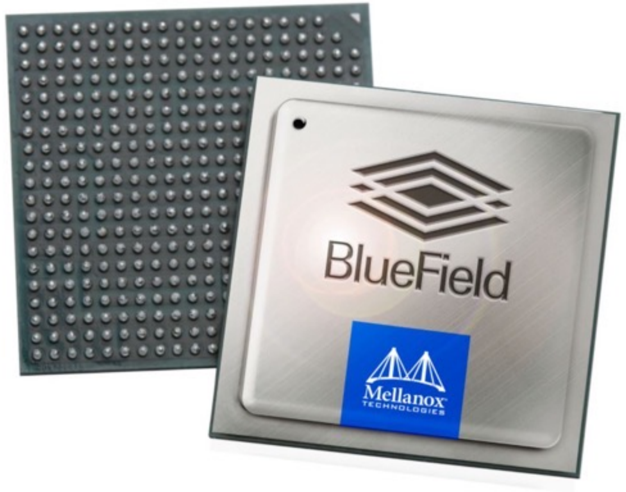
- Amazon, Facebook, Google, Microsoft servers
- Run online services: search, social media, e-commerce
- Dozen cores/server, ~100 W
- Datacenter power ~10-300 MW
- Shared memory (within a single server)
- Price ~ millions-billions CHF

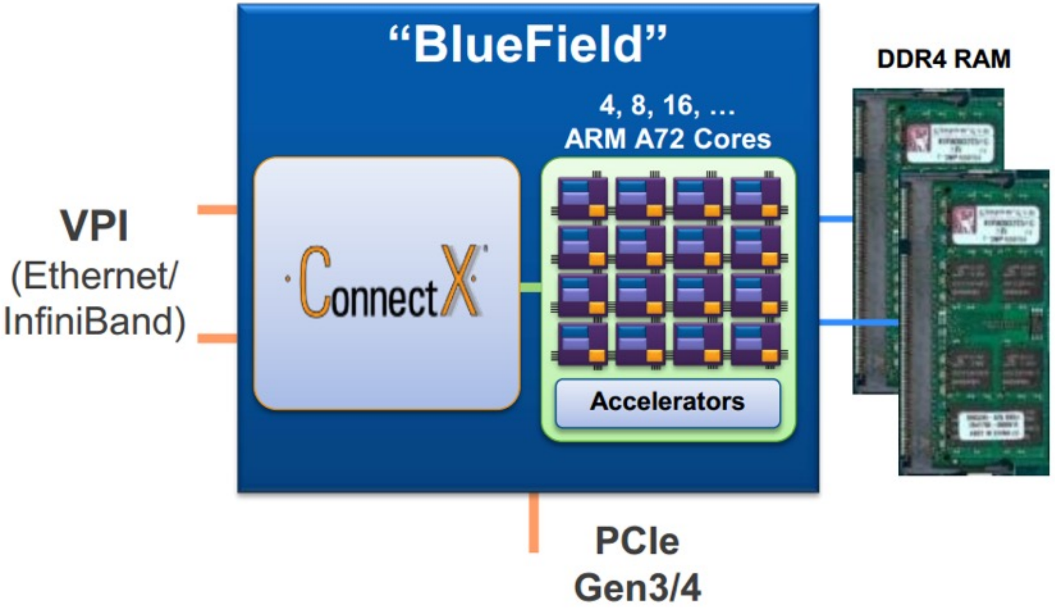


Example: Network Interface Cards

Cache-coherent Manycore

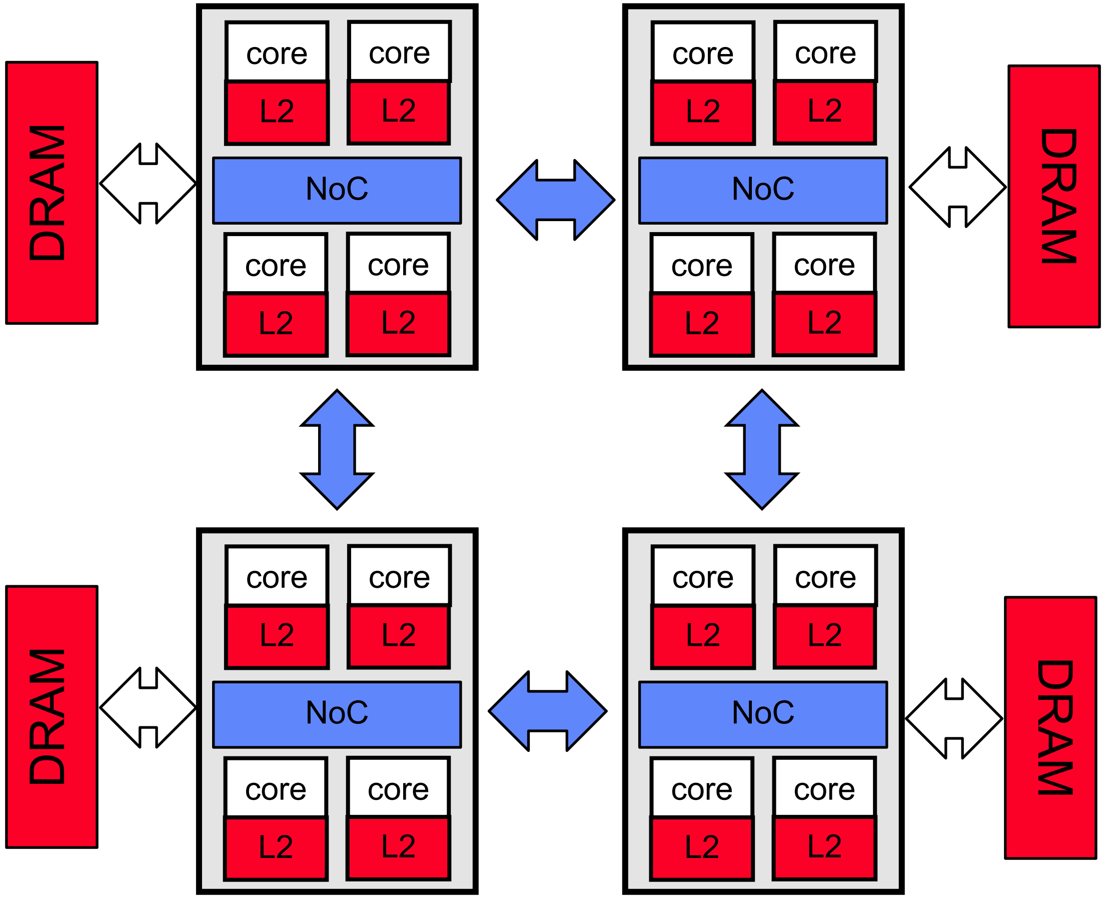
- Up to 32 OoO ARM cores
- Up to 0.5 TB of DRAM
- Directly connected to network





Modern servers are NUMA (non-uniform access)

- Multiple cores/chip
- Multiple levels of memory
- Multiple levels of network



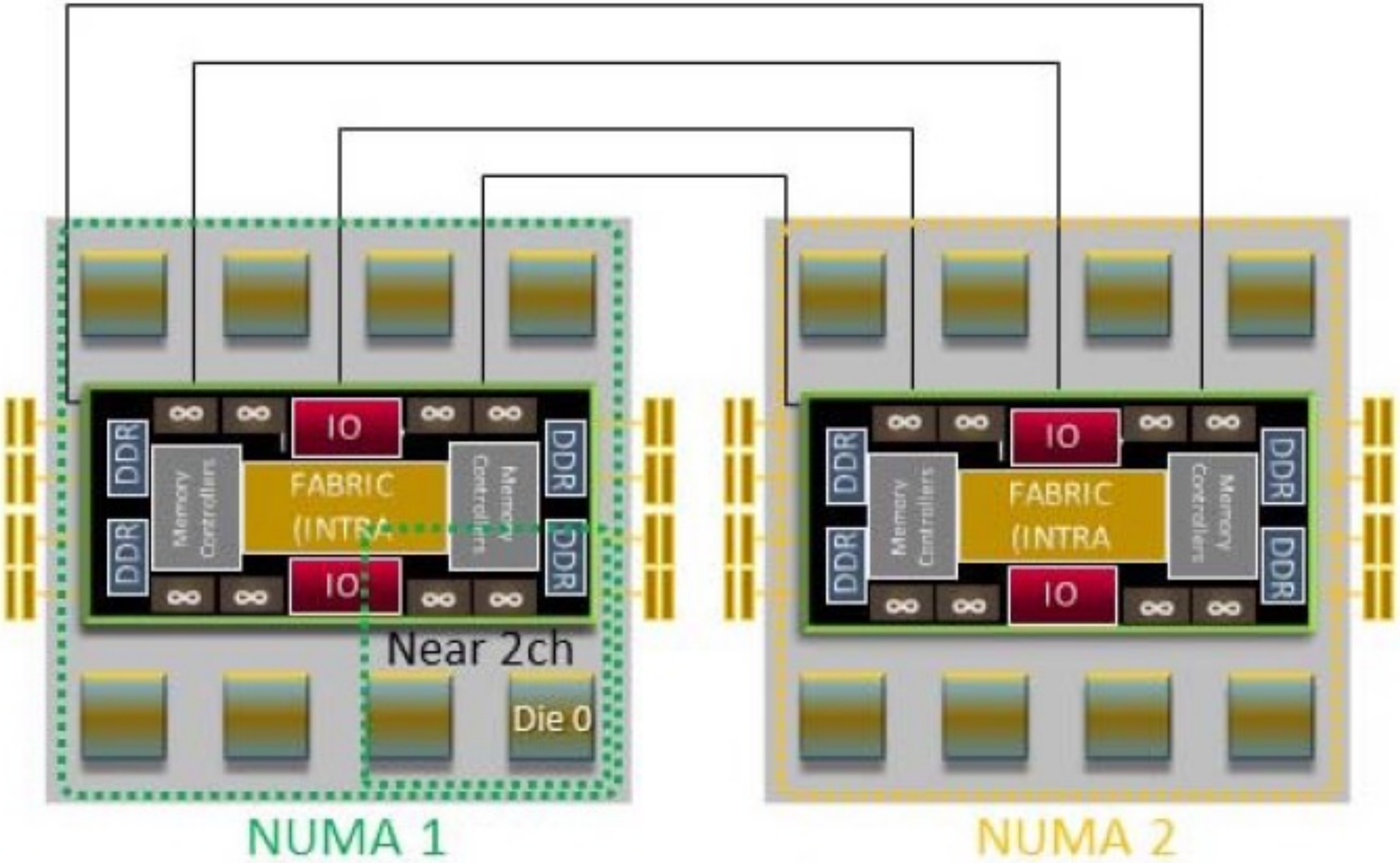
NUMA systems

- Non-uniform latency to memory
  - First-level cache ~ 2 cycles
  - Second-level cache ~ 10 cycles
  - Third-level cache ~ 50 cycles
  - Off-chip own DRAM ~ 150 cycles
  - Off-chip others’ DRAM ~ 300 cycles

- Contention everywhere
  - On-chip network
  - Off-chip network
  - Cache ports, memory channels and cache/memory controllers

Data placement is everything!

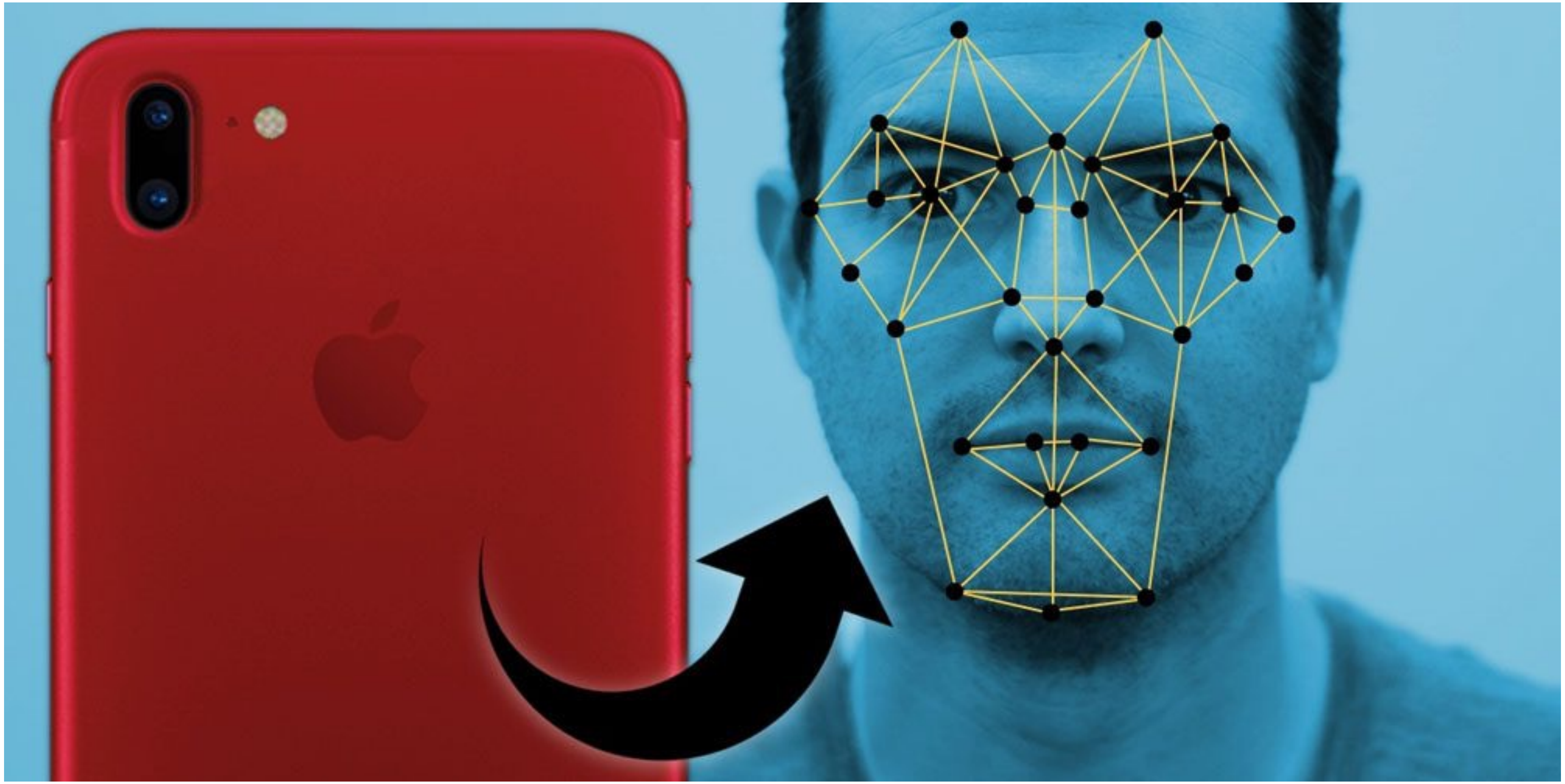
AMD Zen 3 SERIES



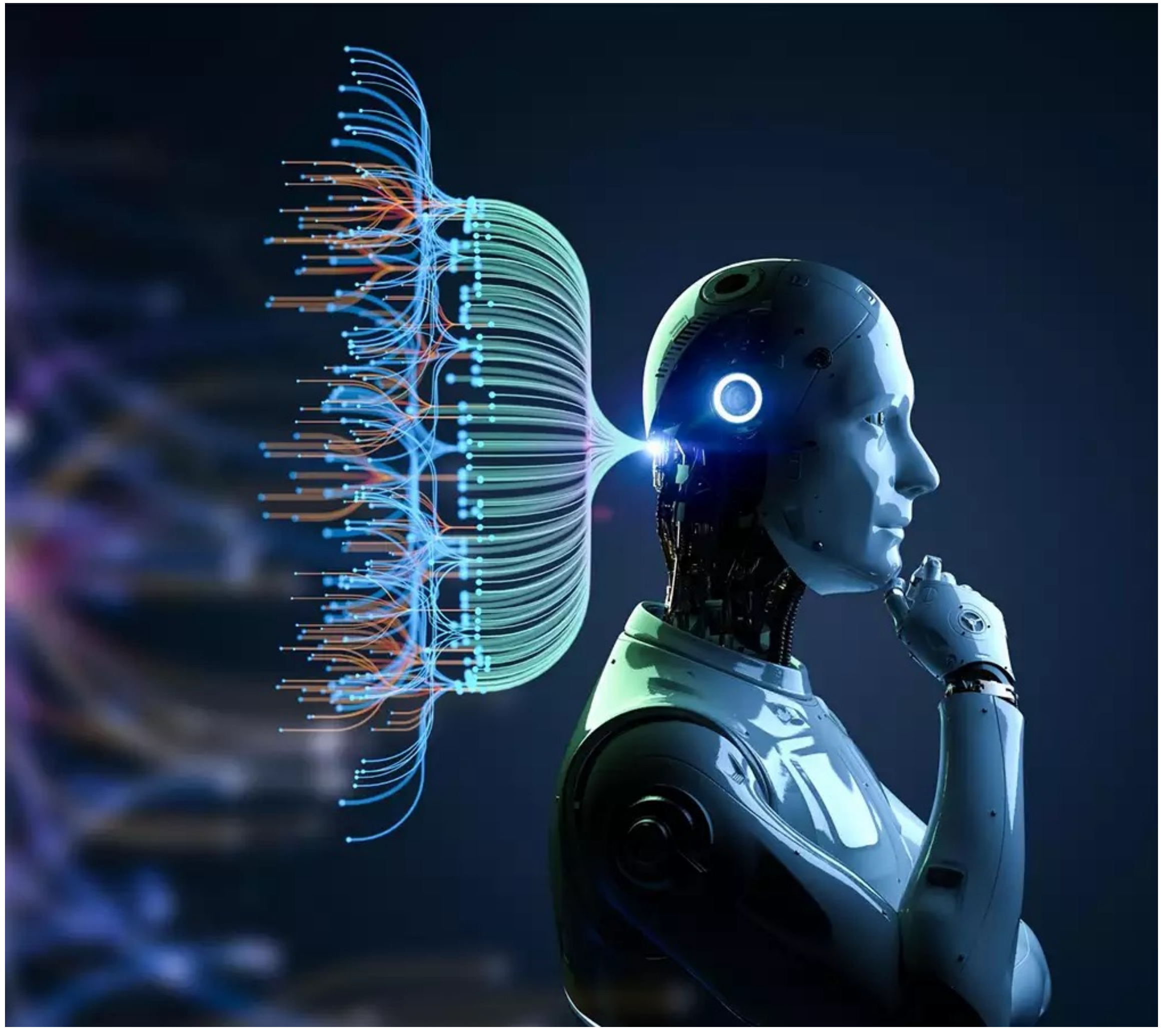
Example: “LUMI”, Finnland

362K CPU cores, 10K GPUs, 8.5MW, top 5 max performance

Example Parallelism: Face Recognition

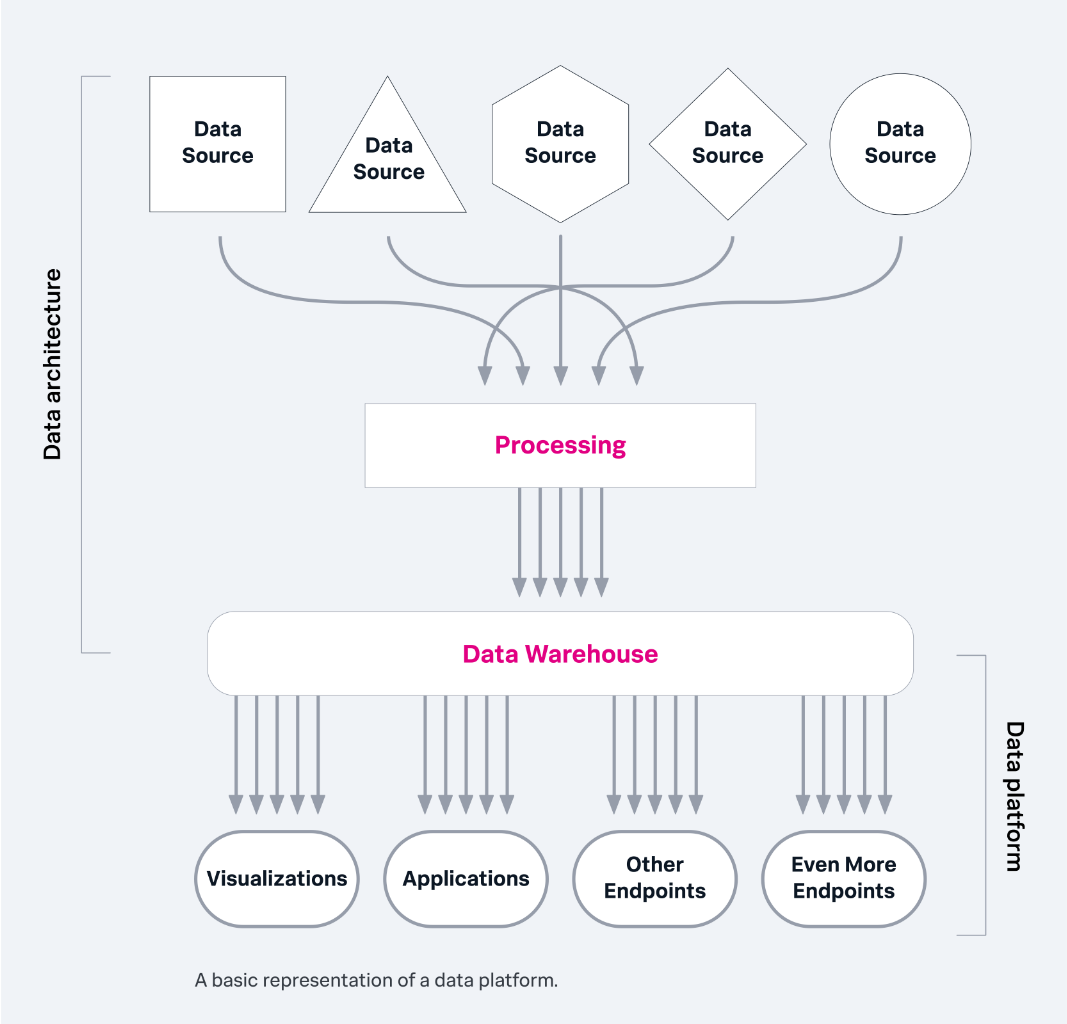


Example Parallelism: Generative AI



- Creates a wide variety of data, such as images, video, audio, text and 3D models
- E.g., GPT and Midjourney

Example Parallelism: Data Platform



A basic representation of a data platform.

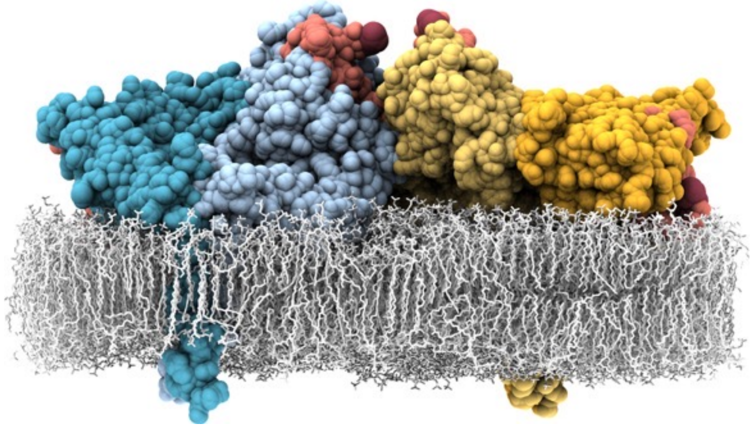
Manage all aspects of data lifecycle including

- Data warehousing
- Analysis
- Processing
- Presenting

Example Parallelism: Science and Engineering

- Examples
  - Weather prediction
  - Drug development
  - Oil reservoir simulation
  - Automobile crash tests
  - ….

Molecular dynamics
used in drug discovery



- Typically, model physical systems or phenomena
- Problems are 2D or 3D
- Usually requires heavy arithmetic

Technical Roadmap for the Course

- Multicores & Multiprocessors
- Parallel Programming Models
- Cache Coherence
- Software Optimizations
- Memory Ordering
- Synchronization
- Concurrent Programming
- GPUs

Hardware Jargon

- Processor & core used interchangeably
- Cores run threads
- Each chip has multiple cores
- Cores may be heterogeneous
  - CPU cores vs. GPU cores vs. accelerators
- Each board can have multiple chips
- Each platform can have multiple boards
  - Mobile platforms usually a single chip/single board
  - Datacenters 10’s to 100’s of thousands of boards

Where are we going?

- How do we execute a program in parallel?
- How do we make sure cores see one copy of memory?
- How do we implement synchronization?
- What is inside a GPU?
- How to program a GPU?
- What happens when we go beyond one chip?

Models of Parallelism

- Instruction-level parallelism
  - Multiple instructions executing concurrently
  - Limited by control and data dependences
  - e.g., Apple A8
- Task-level parallelism
  - Multiple threads executing concurrently
  - Limited by the algorithm and sync. overheads
  - e.g., Amazon Graviton 3
- Data-level parallelism
  - One instruction operating concurrently on different data
  - Limited by irregular data access patterns & memory BW
  - e.g., NVIDIA Titan V

Review: Sources of Cache Misses

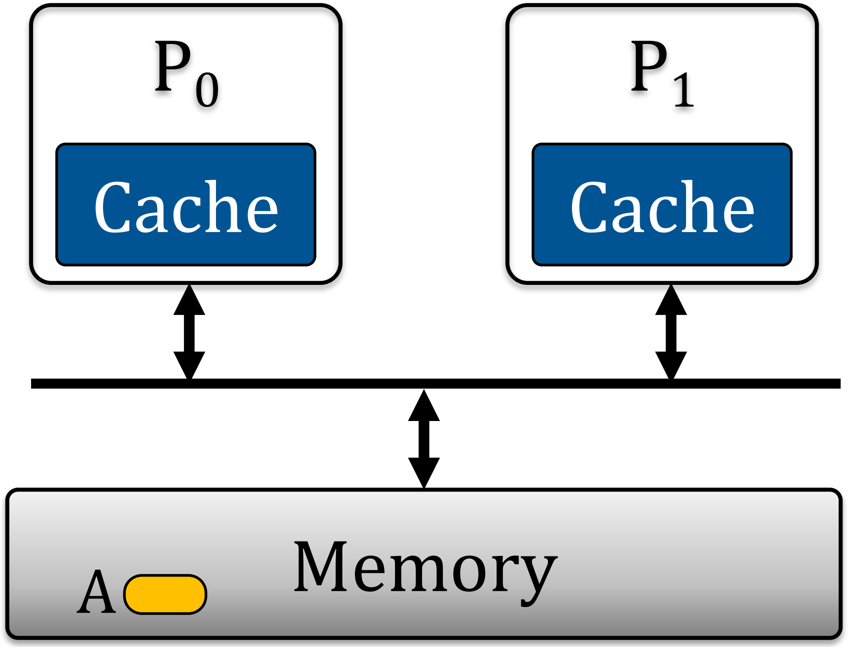
- From CS-200
  - 3 C’s: Compulsory, conflict, capacity misses
- 4th type of miss:
  - Coherence: reads/writes from multiple processors

Cache Coherence

- Needed when processors access shared data
- Reasonable expectation for shared memory
  - Reading address X, should return last value written to address X
  - **What if multiple processors update address X?**
- Coherence provides a global view of address X

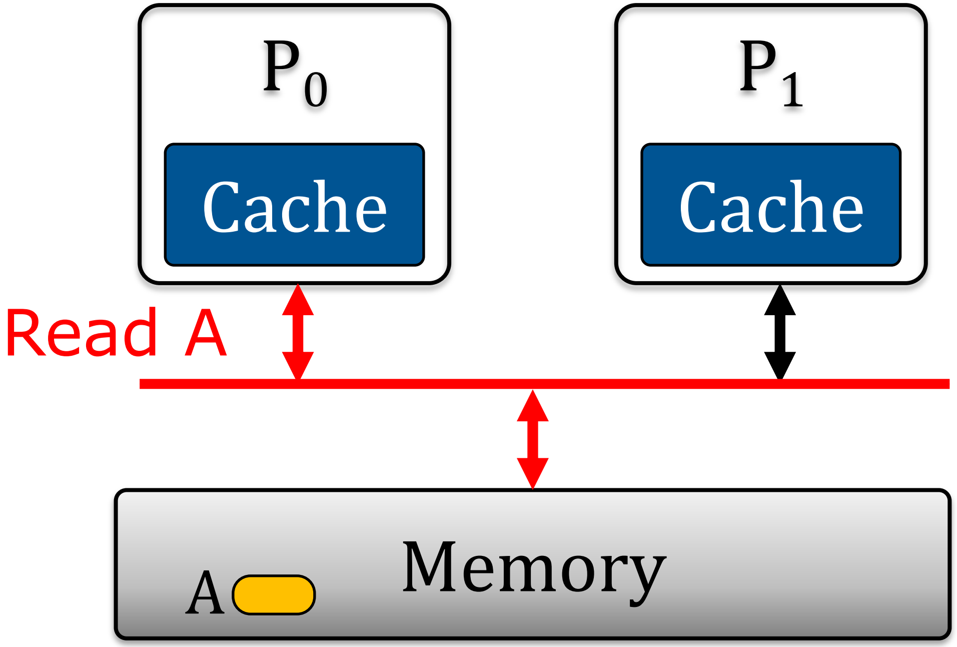
Example: 2 processors reading/writing

- Assume two processors
  - Each has a private cache
- Globally shared interconnect (bus)
- Variable A
  - Starts in memory



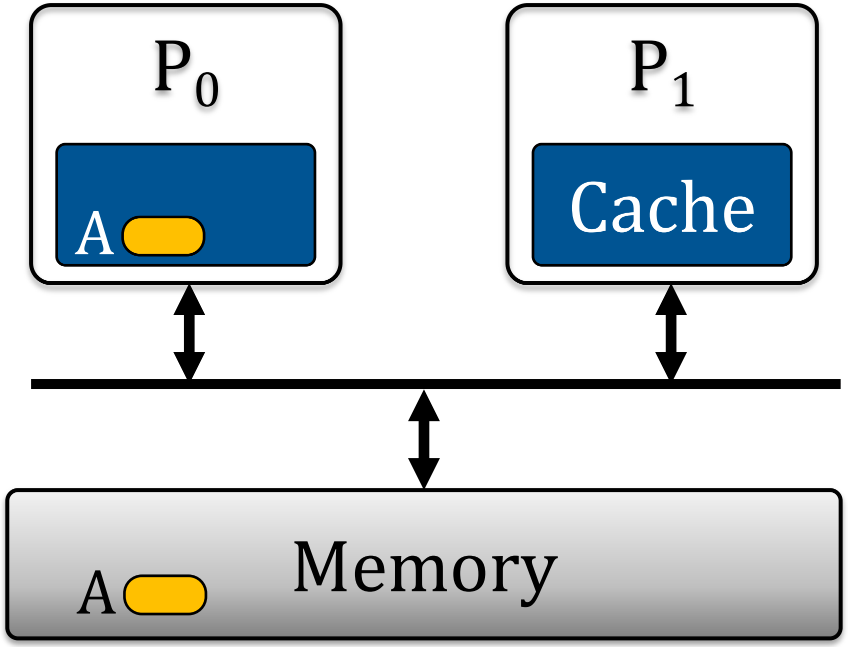
Example: 2 processors reading/writing

1. P_0 - Read A



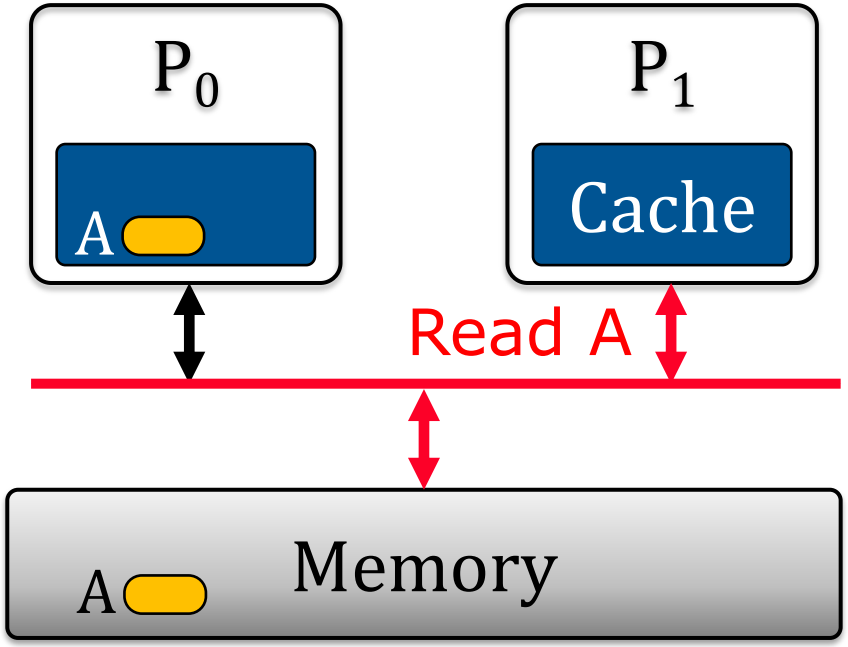
Example: 2 processors reading/writing

1. P_0 - Read A
   a) A now cached by P_0



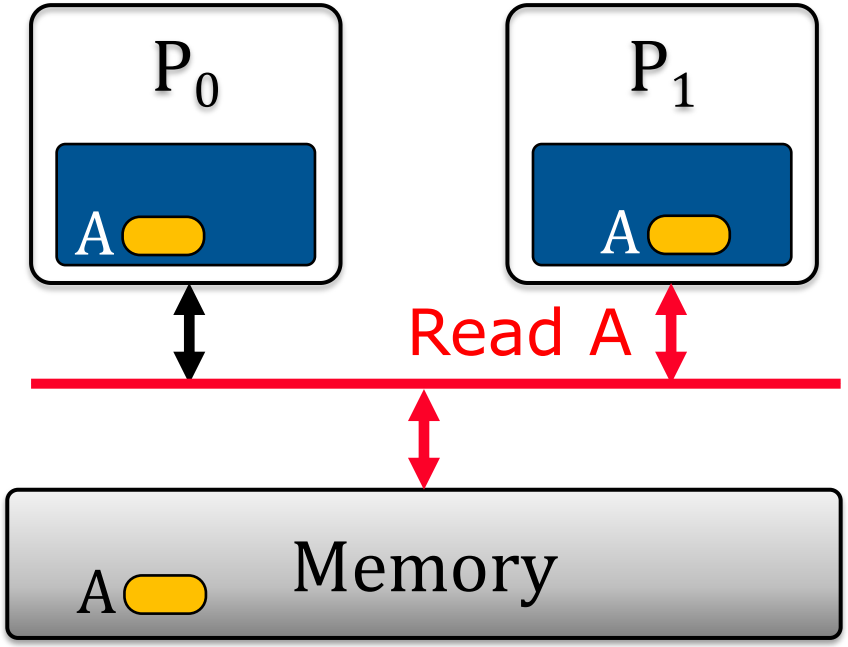
Example: 2 processors reading/writing

1. P_0 - Read A
   a) A now cached by P_0
2. P_1 - Read A



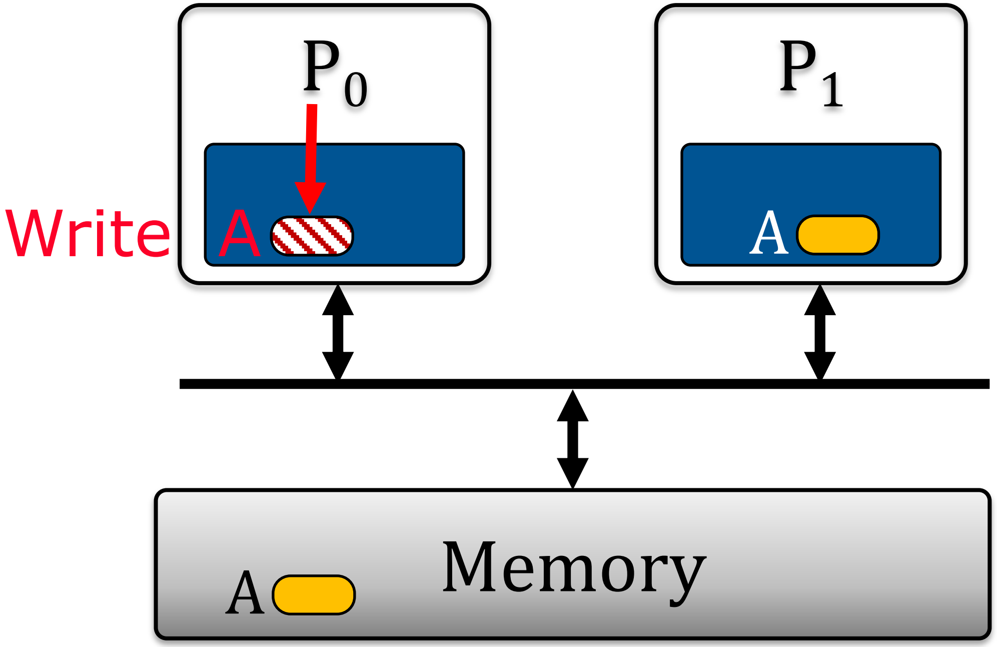
Example: 2 processors reading/writing

1. P_0 - Read A
   a) A now cached by P_0
2. P_1 - Read A
   a) A now cached by P_1



Example: 2 processors reading/writing

1. P_0 - Read A
   a) A now cached by P_0
2. P_1 - Read A
   a) A now cached by P_1
3. P_0 – Write A
   a) Oops!!!!!
   b) A has a different value now in the two caches
   c) Must first get rid of the copy in P_1's cache
   d) Then write can succeed



CS-200 → CS-302

- Learned how to build a uniprocessor cache

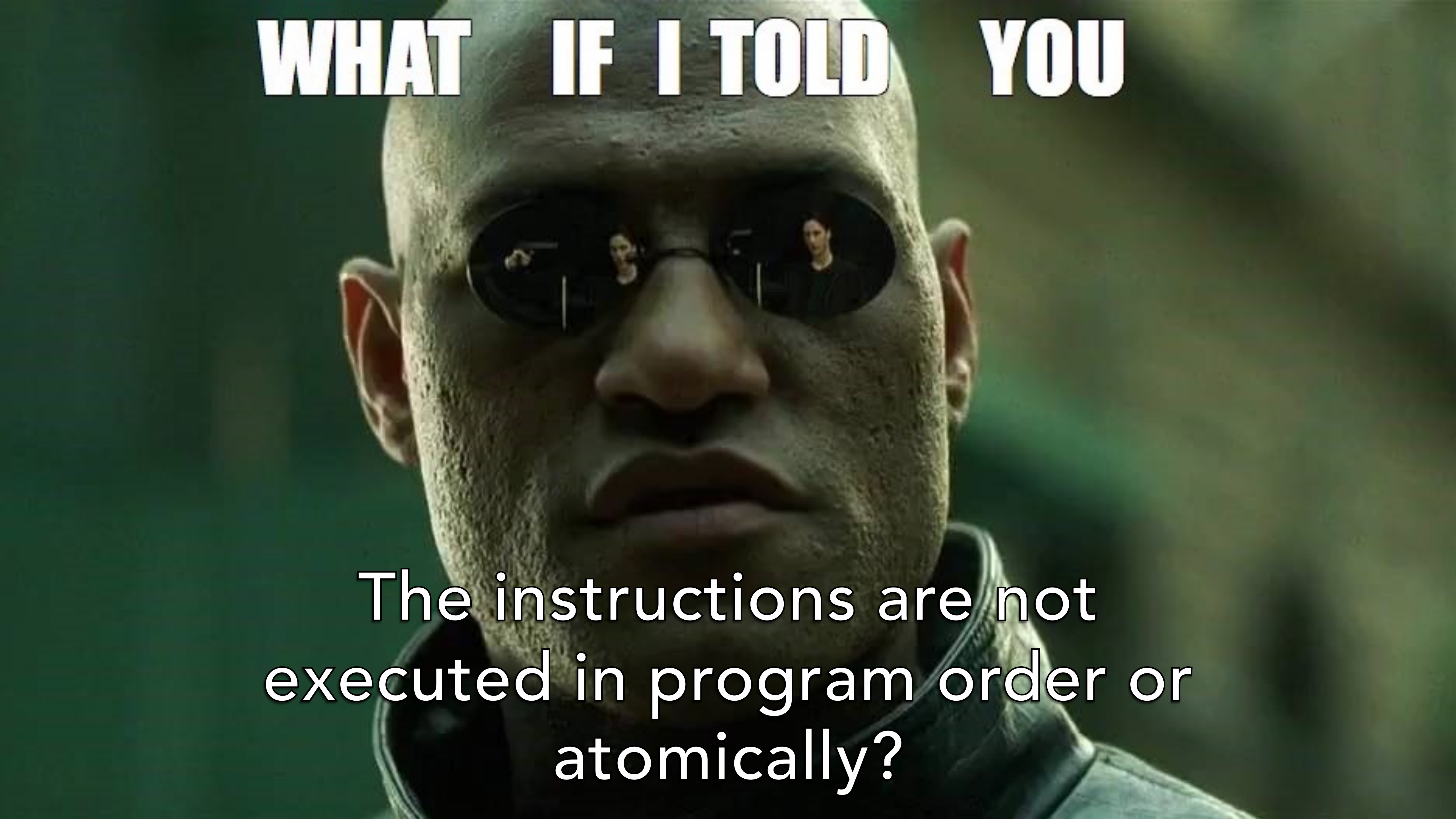
- In this course, you will learn:
  - How memory behaves with multiple cores & caches
  - How to optimize SW in presence of cache & coherence

Memory Ordering

- Cache coherence makes sure all copies of one address have the same value

- But, what happens when multiple addresses are involved
  - E.g., variable X is a lock for the data in variable Y

- Programmers often assume that reads/writes to memory are **in program order** and **atomic** (they start and complete immediately before another access)



The instructions are not executed in program order or atomically?

Memory Ordering

- Loads/stores are not atomic
  - Stores are written in buffers and not yet visible by other cores
  - Caches allow multiple outstanding misses

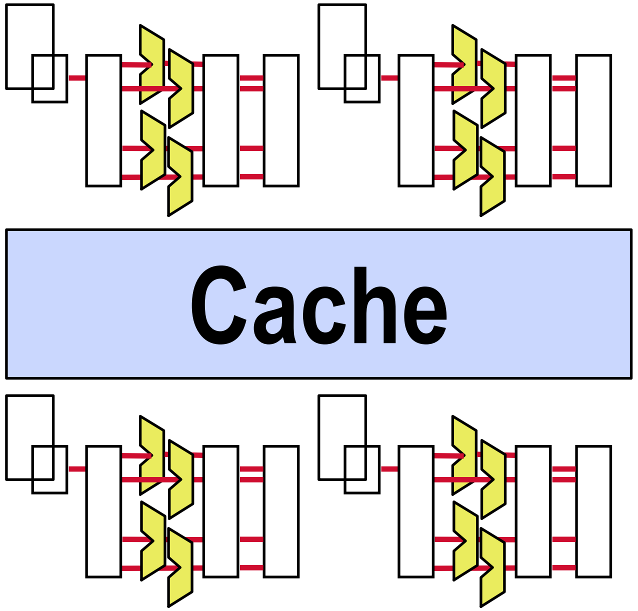
- Instructions are not executed in program order
  - Cores execute out of order
  - Compiler moves instructions around

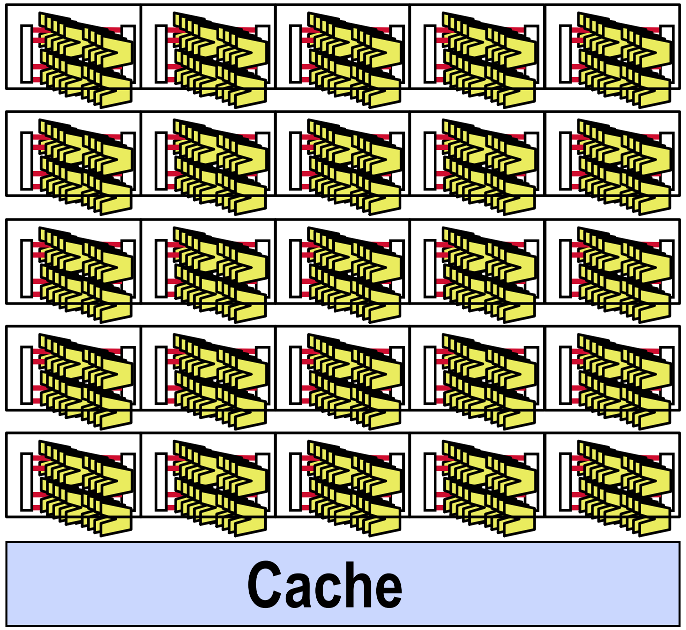
- In CS-206
  - Covered the Java Memory Model and “happens before” relations
  - In this course, we will learn about consistency at the ISA level
  - Also, how software must be aware of the ISA’s memory model

Synchronization

- In CS-214 we learned about concurrent data structures
- In CS-302 we will learn about a spectrum of synchronization primitives and how to build them
- We will learn about language-level all the way down to instruction-level primitives

CPU vs. GPU

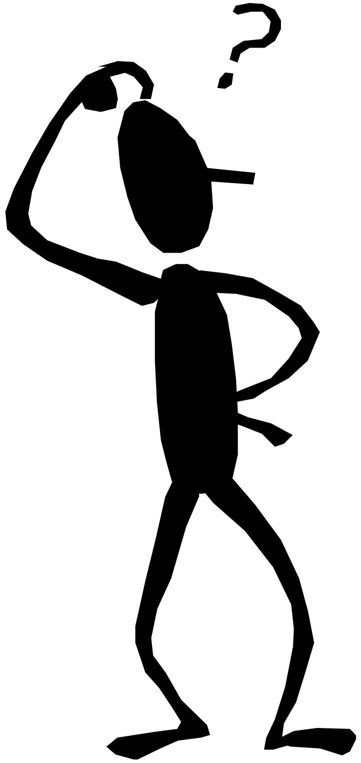




- Tens of cores
- Mostly control logic
- Large caches
- Regular threads (e.g., Java)

- Thousands of tiny cores
- Mostly ALU
- Little cache
- Special threads (e.g., CUDA)

Feedback, Please!



- Ask, don't struggle when you don't grasp a new concept!

  You are here to learn and we to explain…

- Give immediate feedback if the background is new to you:

  "We have no idea what you are talking about! We have never seen this stuff!"

- Contact us if you have more elaborate feedback: e-mail first and a meeting later if needed

Summary

- Parallel & Concurrent Software
  - Parallel computing
  - Shared memory
  - Message passing
  - Memory consistency
  - Synchronization
  - Continuations/RPC
  - Multithreading
  - GPUs
  - Beyond GPUs

- Learn about hardware design tradeoffs with parallelism in mind