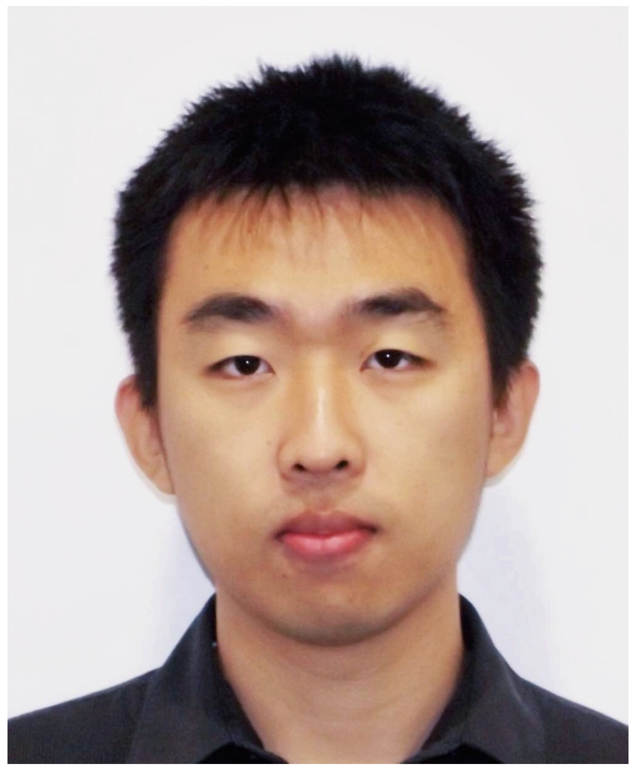
Lecture 1:

Why Parallelism?
Why Efficiency?

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Fall 2023

Hi!

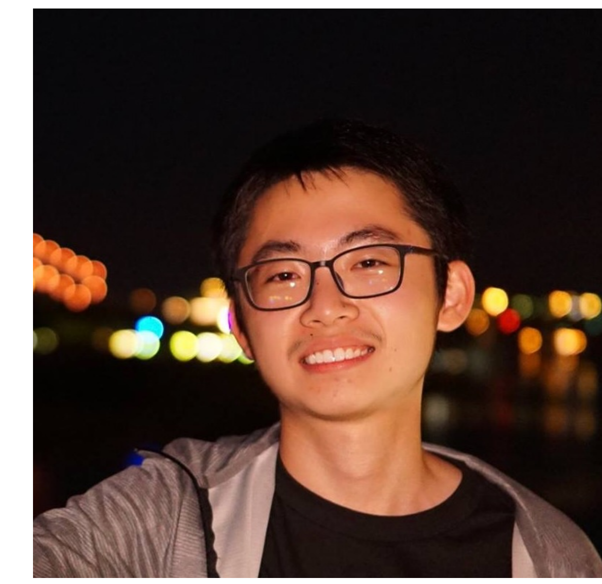


Prof. Jia

Prof. Skarlatos

Shreya

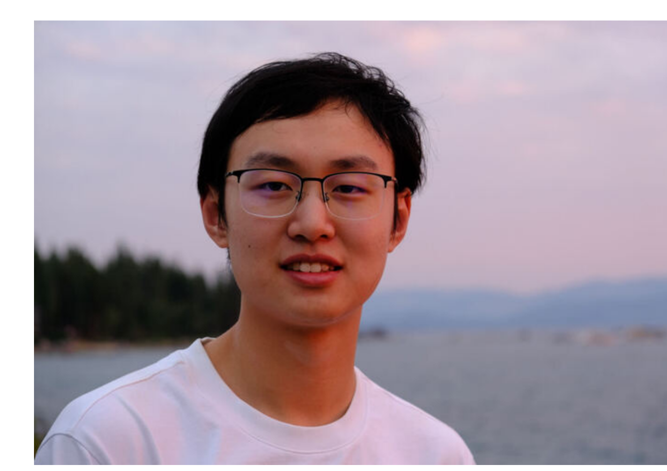
Fanyue



Kaiyang

Rene

Mingkang



Zikun

What will you be doing in this course?

Programming Assignments

- **Four programming assignments**
  - **First assignment is done individually, the rest will be done in pairs**
  - **Each uses a different parallel programming environment**

Assignment 1: ISPC programming on Intel quad-core CPU (and Xeon Phi)

Assignment 2: CUDA programming on NVIDIA GPUs

Assignment 3: Parallel Programming via a Shared-Address Space Model

Assignment 4: Parallel Programming via a Message Passing Model

If you are on the Wait List

- We will hand out Assignment 1 later this week
- Our algorithm for filling the K remaining slots in the class:
  - the first K students on the Wait List who hand in Assignment 1 and receive an A on it are enrolled in the class

Exams

- We will have two midterm-style exams
  - Each covers roughly half of the course material
- No final exam
  - We use the final exam slot for our project poster session

Written Assignments

- We will have roughly 3 written assignments
  - Peer graded

- No programming is involved
  - They are paper-and-pencil type problems

- Purpose:
  - Practice answering exam-style questions

Final project

- 6-week self-selected final project
- Performed in groups (by default, 2 people per group)
- Start thinking about your project ideas TODAY!
- Poster session during the final exam slot

- Check out last year's projects:

http://www.cs.cmu.edu/afs/cs/academic/class/15418-f22/www/projects.html

Participation Grade: Online Mini-Quizzes

- During the day of a lecture, we will have a simple quiz posted on Canvas
- The quizzes should be easy
  - the goal is just to demonstrate that you are keeping up with the class material
- They also give us feedback on what the class is understanding

Grades

40% Programming assignments (4)

11% Written assignments + Quizzes

24% Exams (2)

25% Final project

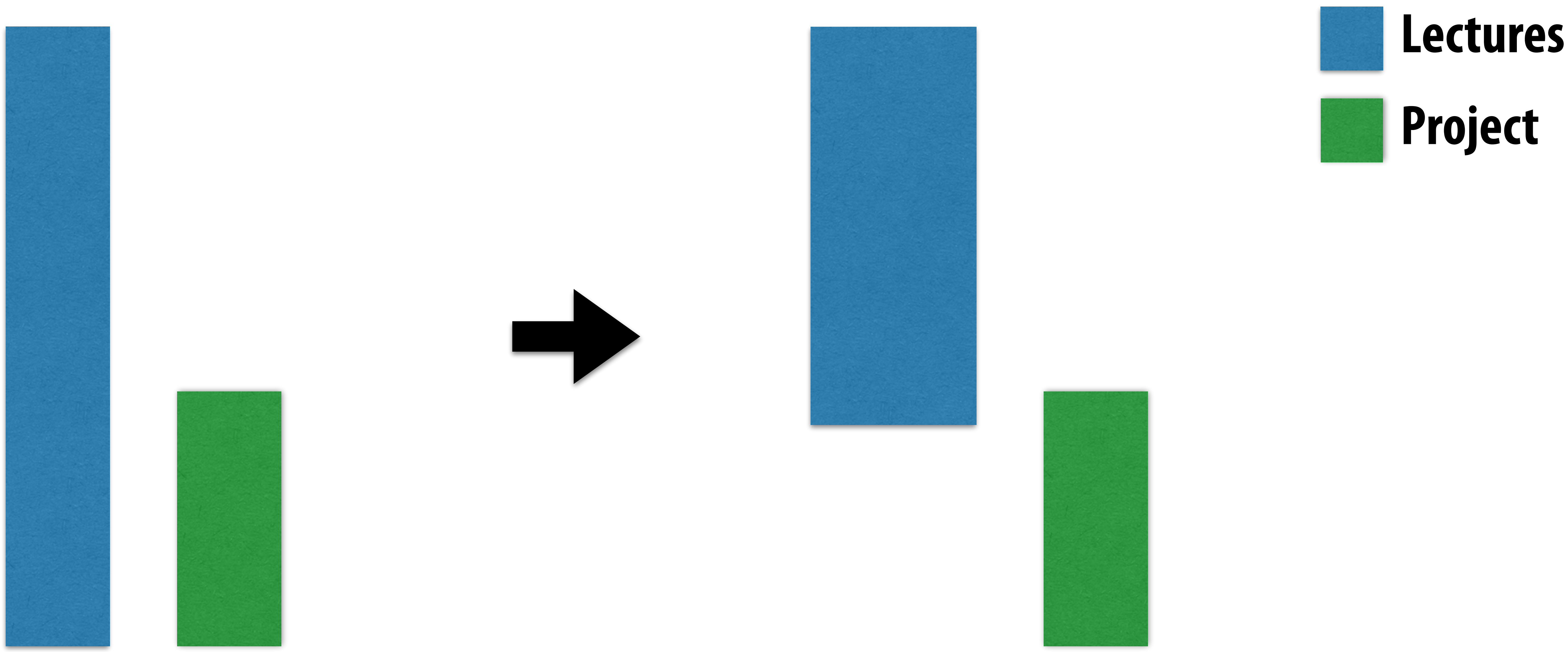
Each student (or group) gets up to five late days on programming assignments (see syllabus for details)

Getting started

- **Course Information:**
  - https://www.cs.cmu.edu/~418
- **Pay attention to Piazza posts**
  - https://piazza.com/cmu/fall2023/15418618/home
- **Textbook**
  - There is no course textbook, but please see web site for suggested references
  - Canvas includes additional lecture notes
- **Find a partner**
  - Assignments 2-4, Final project

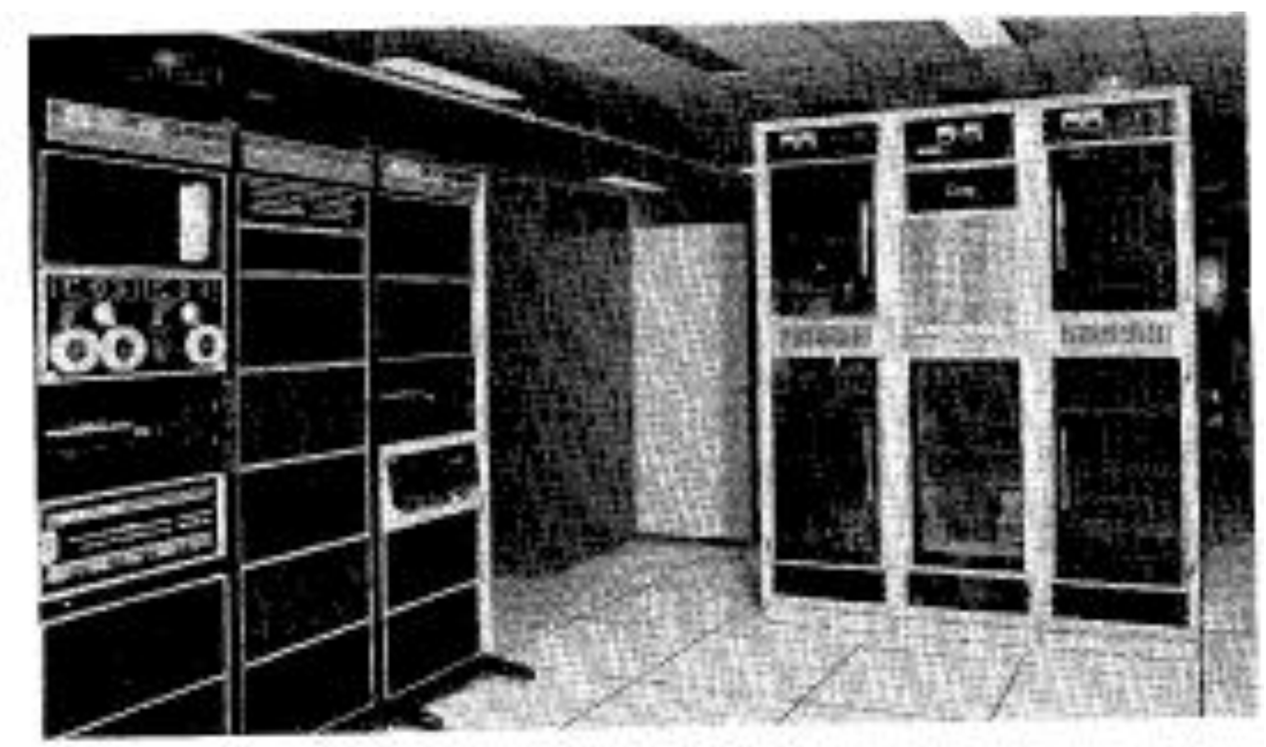
Regarding the class meeting times

- We meet 3 days a week (MWF) for the first 2/3 of the semester
- Same content as 2 days a week over a full semester, but two major advantages this way:
  - you are better prepared to do an interesting project
  - more time to focus on your project



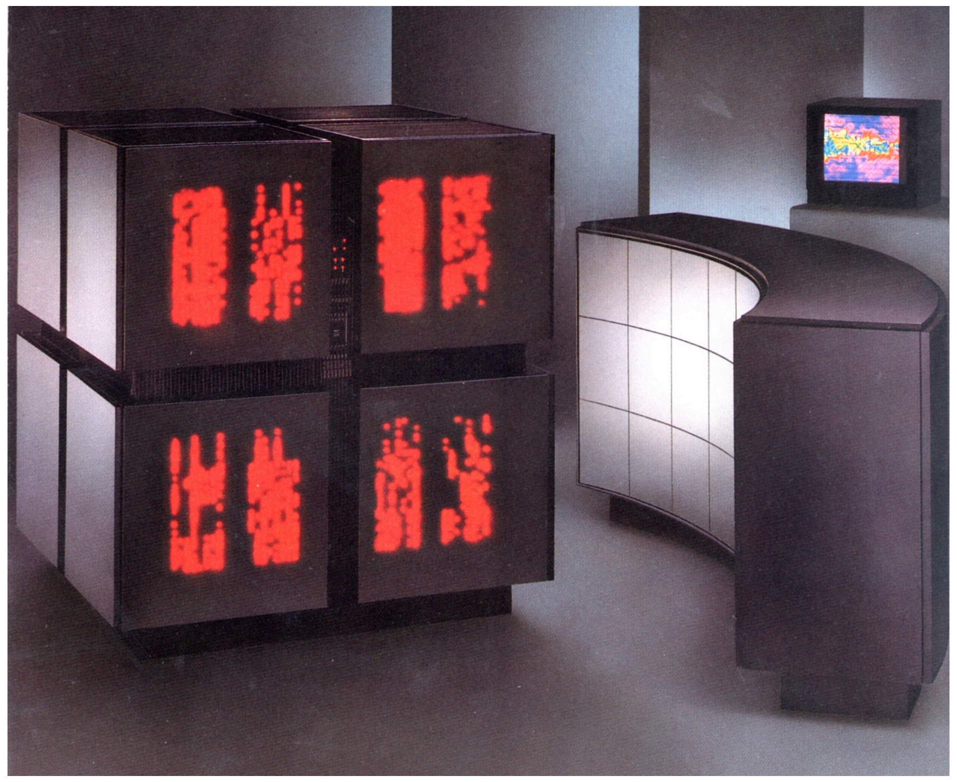
A Brief History of Parallel Computing

- Initial Focus (starting in 1970s): "Supercomputers" for Scientific Computing



C.mmp at CMU (1971)
16 PDP-11 processors

Cray XMP (circa 1984)
4 vector processors



Thinking Machines CM-2 (circa 1987)
65,536 1-bit processors +
2048 floating-point co-processors

← Blacklight at the Pittsburgh Supercomputer Center

SGI UV 1000cc-NUMA (today)
4096 processor cores

A Brief History of Parallel Computing

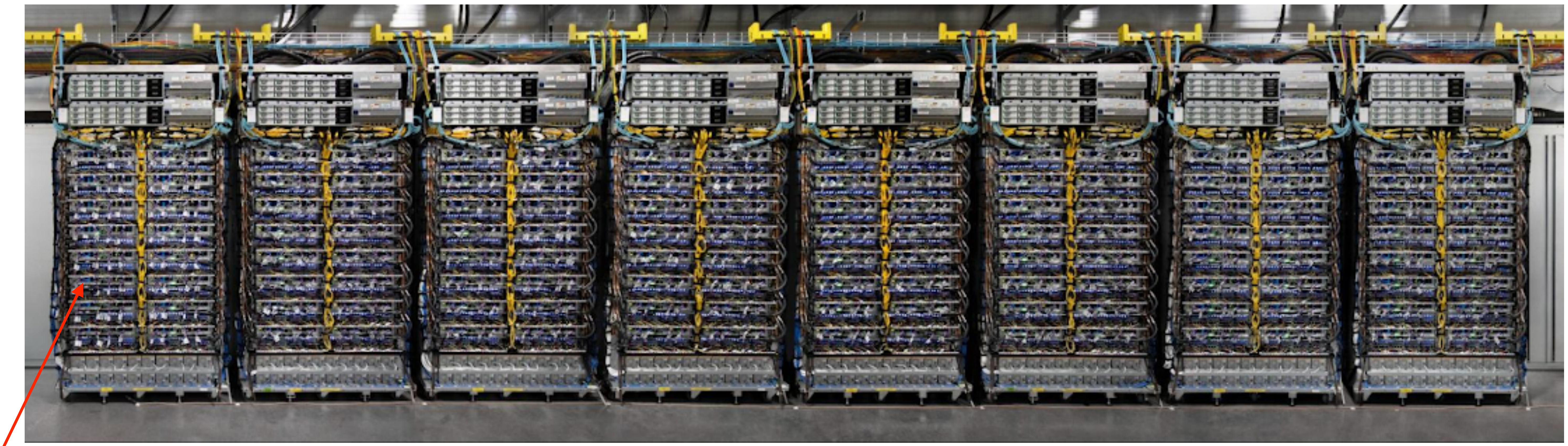
- **Initial Focus (starting in 1970s): "Supercomputers" for Scientific Computing**
- **Another Driving Application (starting in early '90s): Databases**

Sun Enterprise 10000 (circa 1997)
16 UltraSPARC-II processors

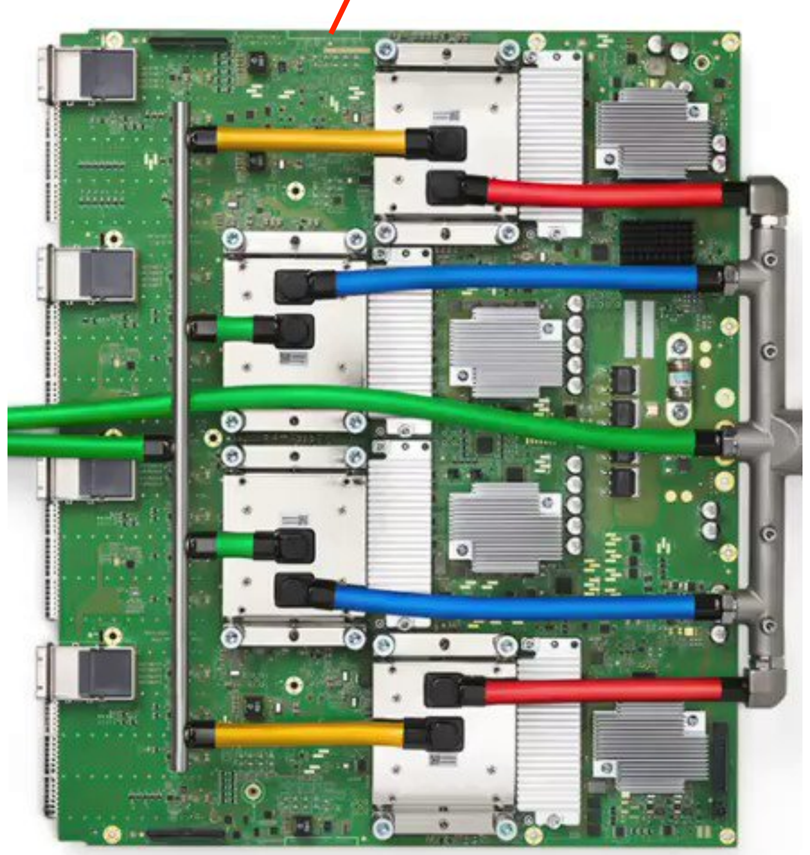
Oracle Supercluster M6-32 (today)
32 SPARC M2 processors

Today: Parallel Computing for ML

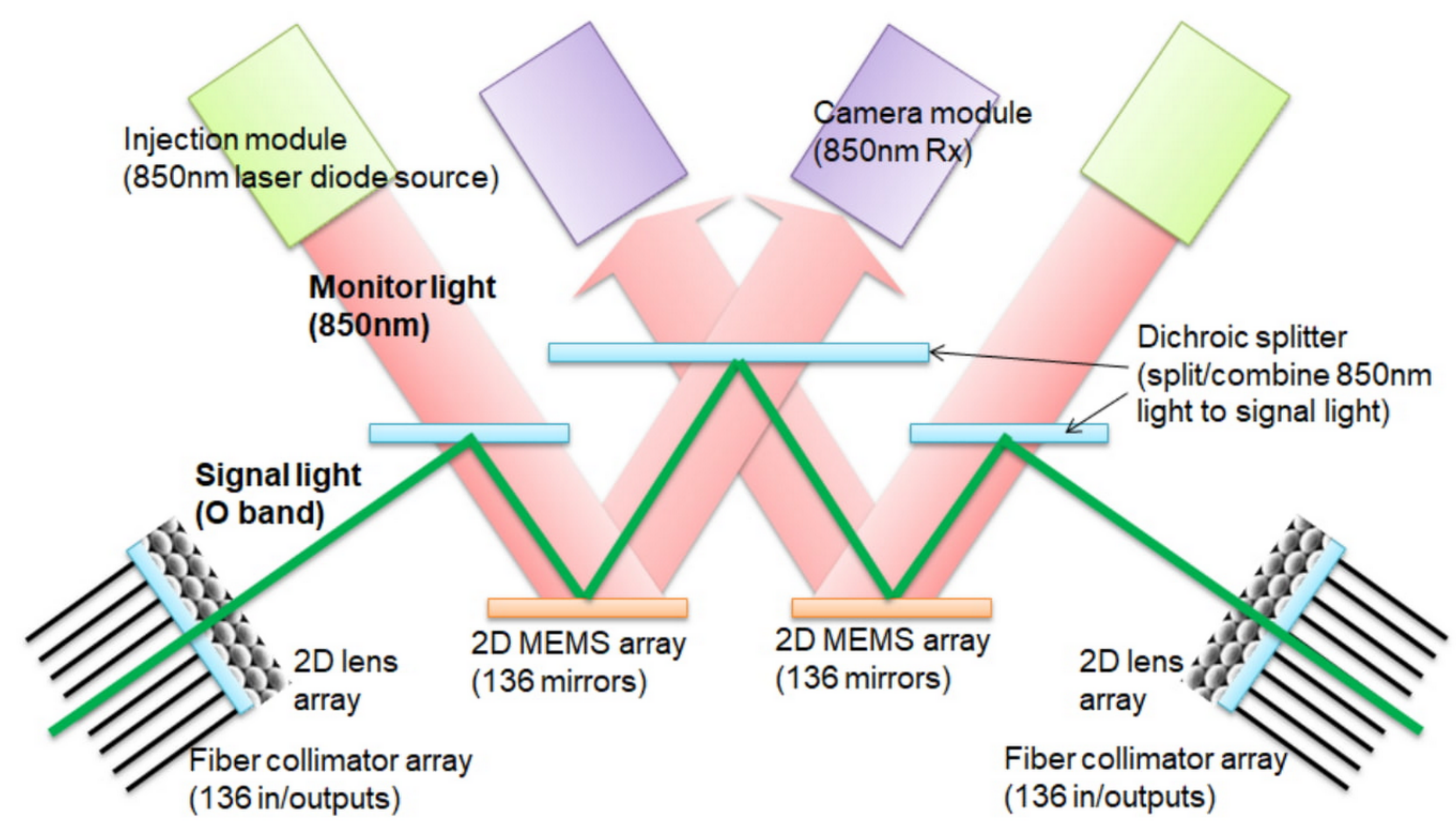
- Current Focus: "Supercomputers" for Machine Learning



TPU v4 Cluster Google (2023)
4096 Chips per Pod



TPU v4 Chip (2023)
275 TeraFlops



Optically-Reconfigurable Network (2023)

Setting Some Context

- Before we continue our multiprocessor story, let's pause to consider:
  - Q: what had been happening with single-processor performance?
- A: since forever, they had been getting exponentially faster
  - Why?

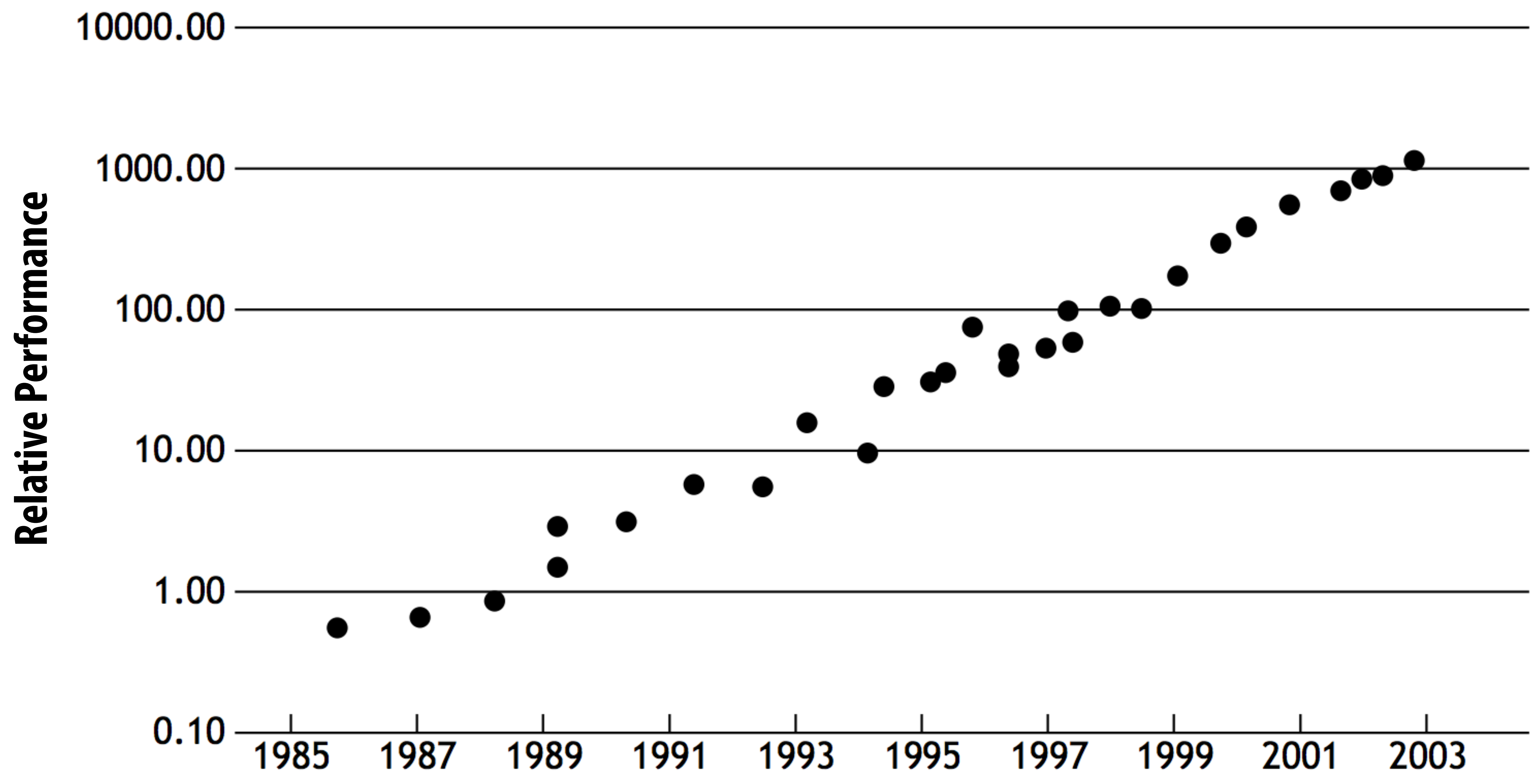


Image credit: Olukutun and Hammond, ACM Queue 2005

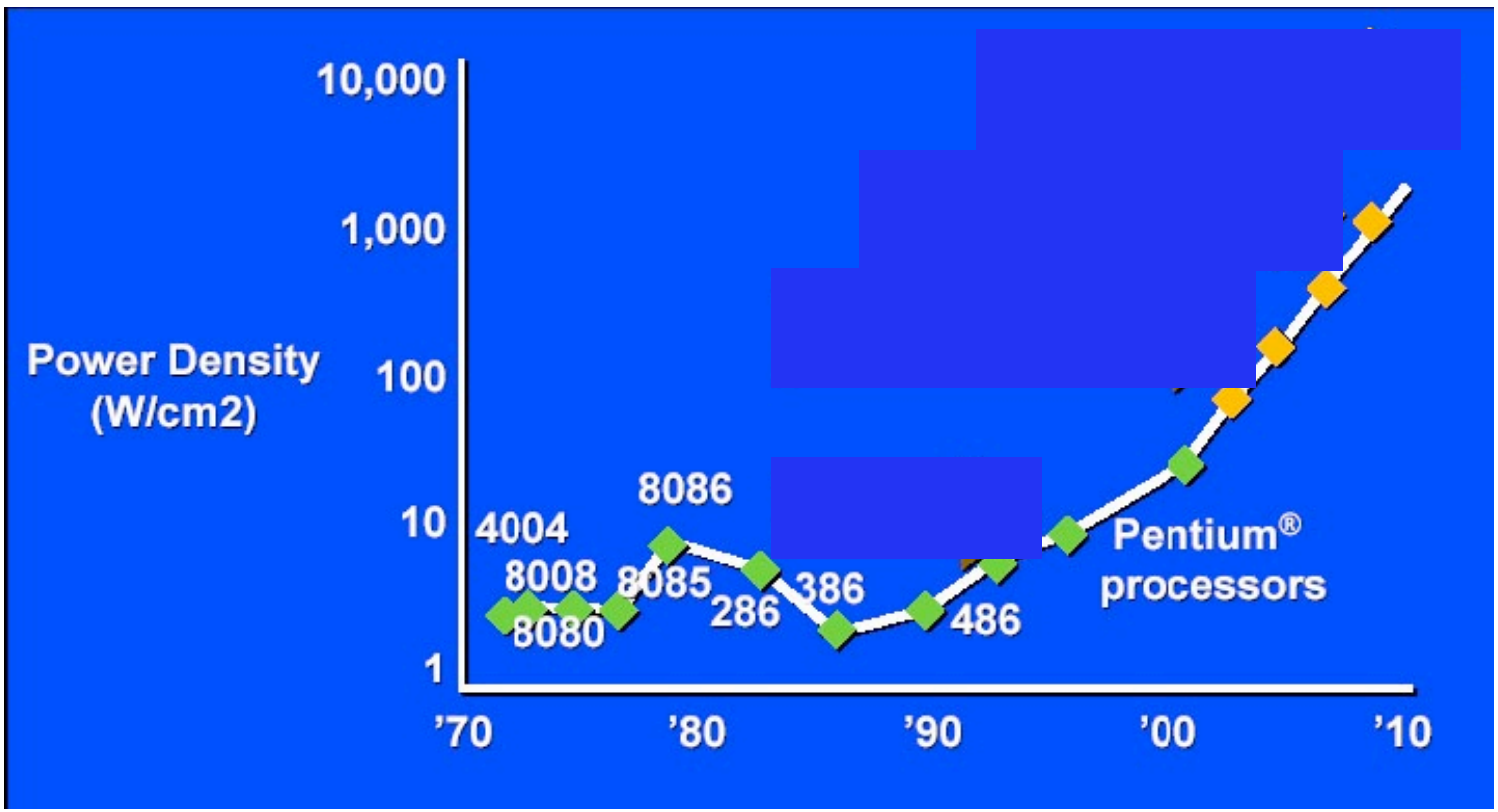
A Brief History of Processor Performance

- **Wider data paths**
  - 4 bit → 8 bit → 16 bit → 32 bit → 64 bit
- **More efficient pipelining**
  - e.g., 3.5 Cycles Per Instruction (CPI) → 1.1 CPI
- **Exploiting instruction-level parallelism (ILP)**
  - "superscalar" processing: e.g., issue up to 4 instructions/cycle
- **Faster clock rates**
  - e.g., 10 MHz → 200 MHz → 3 GHz

- **During the 80s and 90s:** **large exponential performance gains**
  - and then…

A Brief History of Parallel Computing

- Initial Focus (starting in 1970s): "Supercomputers" for Scientific Computing
- Another Driving Application (starting in early '90s): Databases
- Inflection point in 2004: Intel hits the Power Density Wall



Pat Gelsinger, ISSCC 2001

From the New York Times

Intel's Big Shift After Hitting Technical Wall

The warning came first from a group of hobbyists that tests the speeds of computer chips. This year, the group discovered that the Intel Corporation's newest microprocessor was running slower and hotter than its predecessor.

What they had stumbled upon was a major threat to Intel's longstanding approach to dominating the semiconductor industry - relentlessly raising the clock speed of its chips.

Then two weeks ago, Intel, the world's largest chip maker, publicly acknowledged that it had hit a "thermal wall" on its microprocessor line. As a result, the company is changing its product strategy and disbanding one of its most advanced design groups. Intel also said that it would abandon two advanced chip development projects, code-named Tejas and Jayhawk.

Now, Intel is embarked on a course already adopted by some of its major rivals: obtaining more computing power by stamping multiple processors on a single chip rather than straining to increase the speed of a single processor.

…

John Markoff, New York Times, May 17, 2004

ILP tapped out + end of frequency scaling

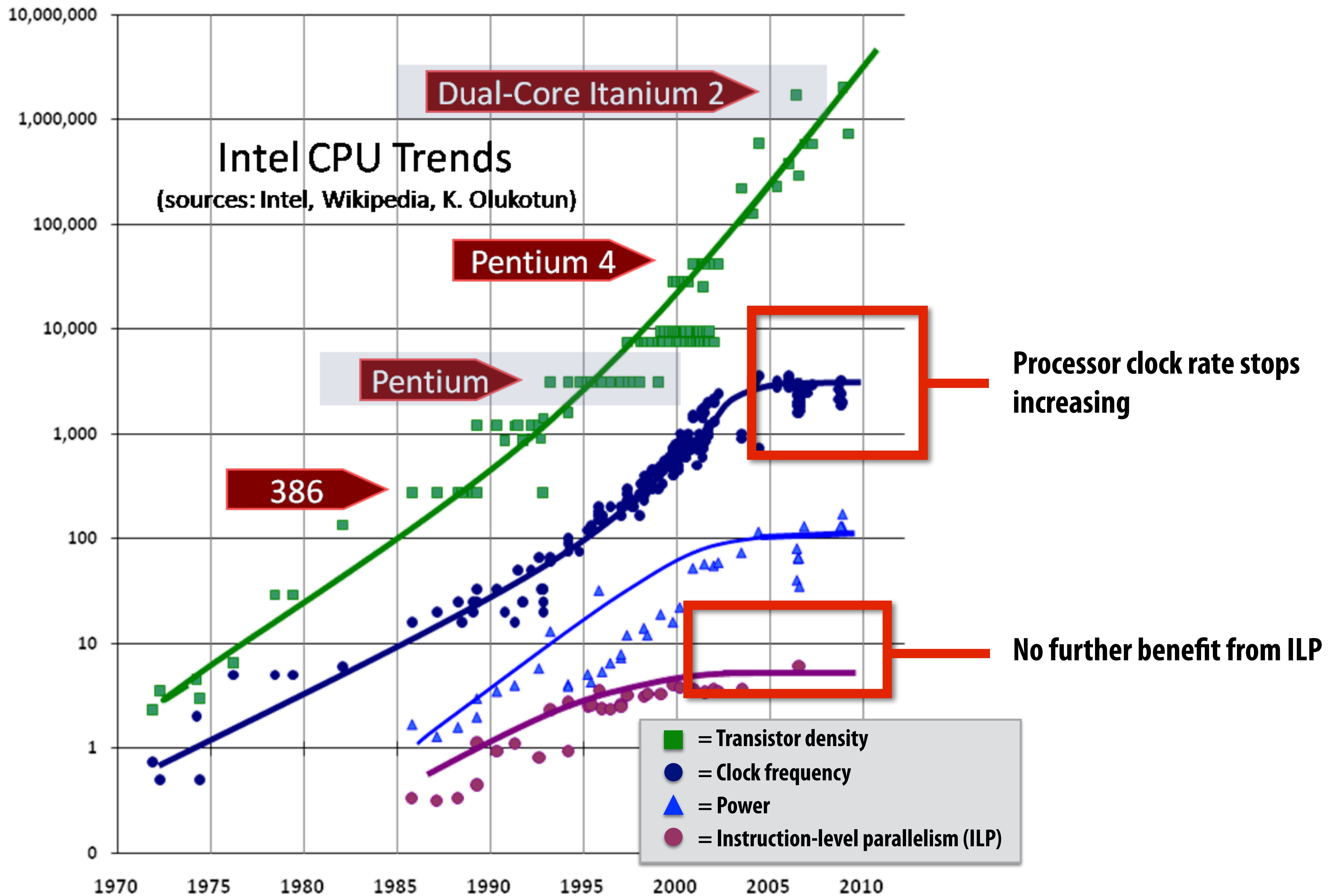


Image credit: "The free Lunch is Over" by Herb Sutter, Dr. Dobbs 2005

Programmer's Perspective on Performance

Question: How do you make your program run faster?

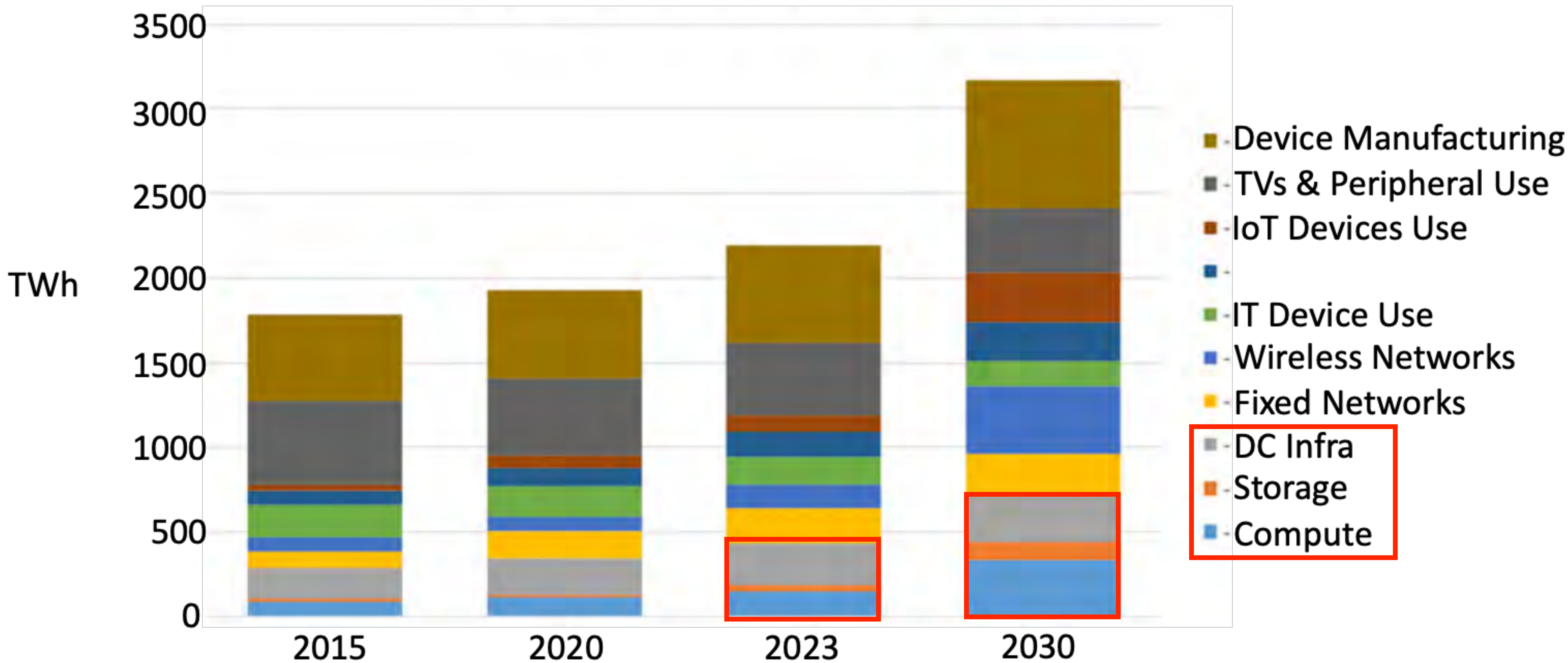
Answer *before* 2004:

- Just wait 6 months, and buy a new machine!
- *(Or if you're really obsessed, you can learn about parallelism.)*

Answer *after* 2004:

- You need to write parallel software.

Power Consumption of Datacenters



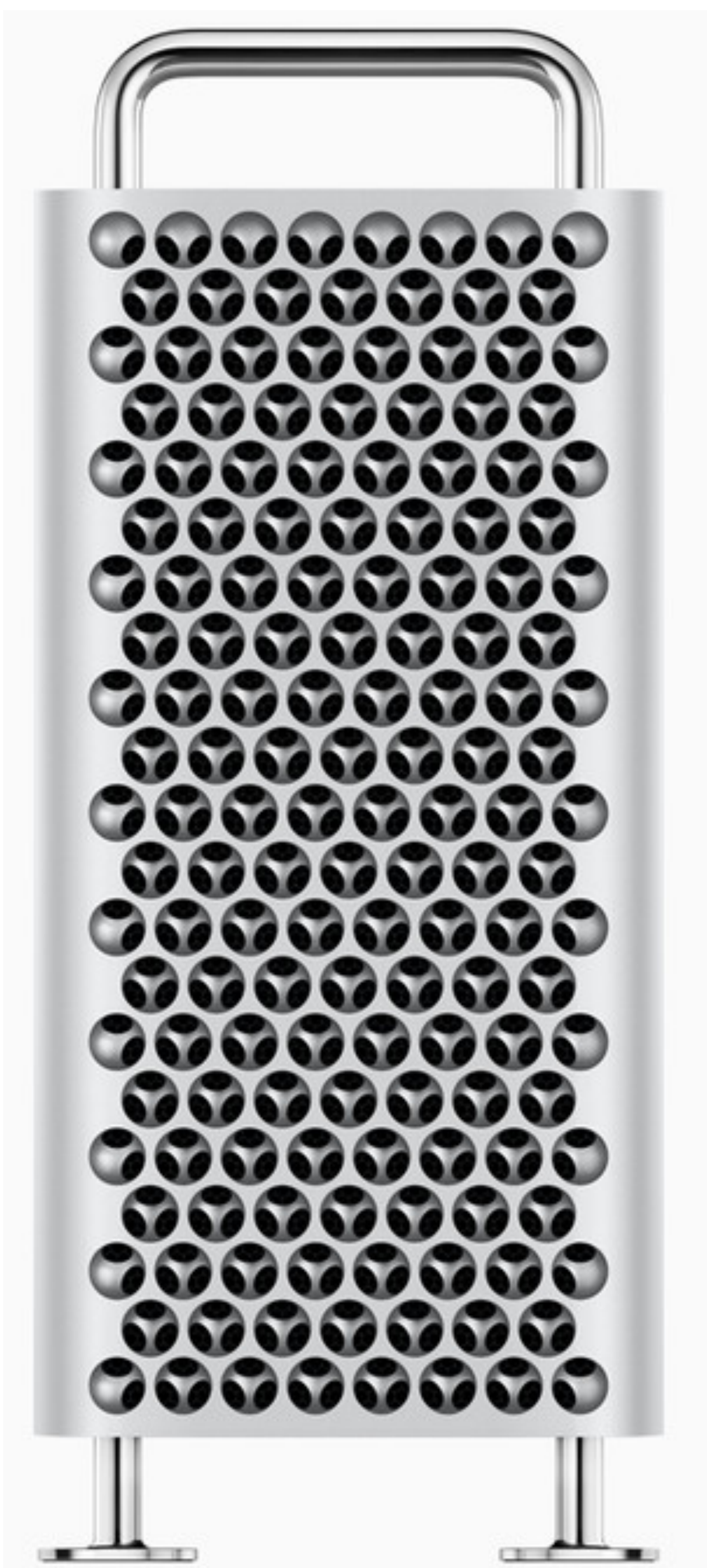
50% Increase!

Life Is On Schneider Electric

Parallel Machines Today

Examples from Apple's product line:

Mac Pro
24 Apple M2 cores



iPad Pro
8 A12X cores
(4 fast + 4 low-power)

iPhone XS
6 A12 cores
(2 fast + 4 low-power)

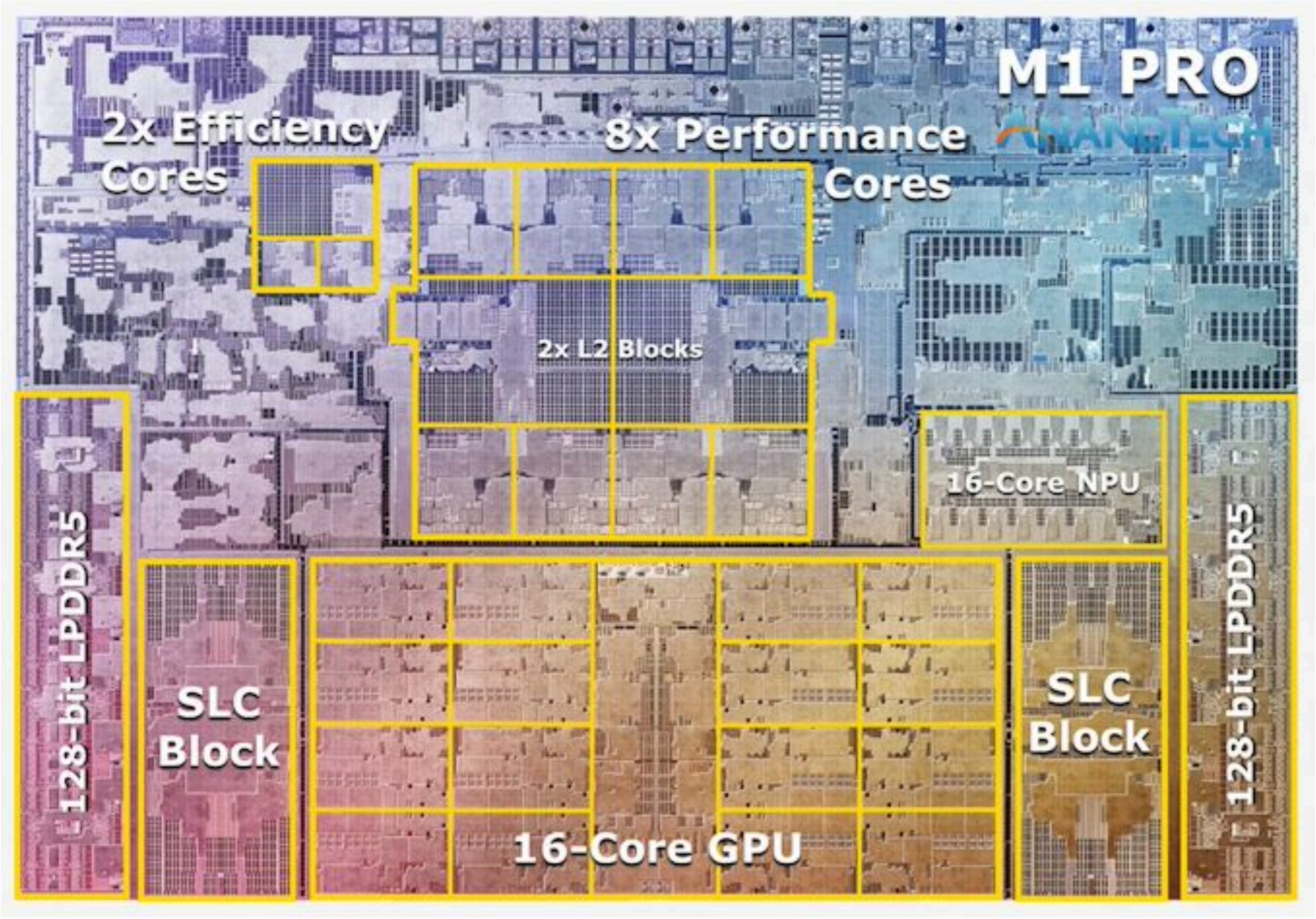
MacBook Pro 14"
12 Apple M2 cores

(images from apple.com)

Intel Alder Lake-S (2021)

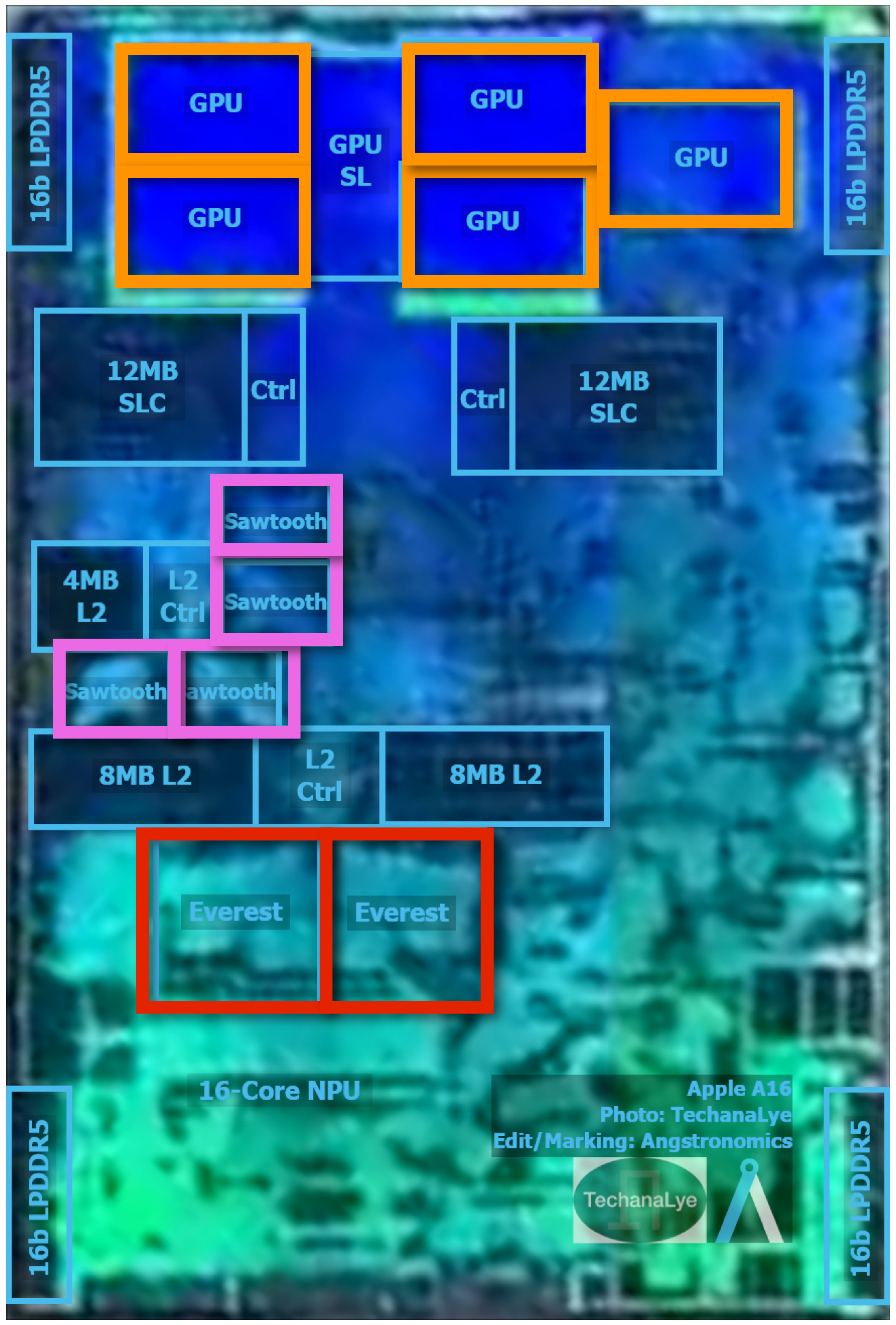
16 CPU cores (8 performance + 8 efficiency)

Laptops: Apple M1 Pro (2021)



- 10 CPU Cores (8 performance + 2 efficiency)
- 16 GPU Cores

Phones: Apple A16 (iPhone 14)

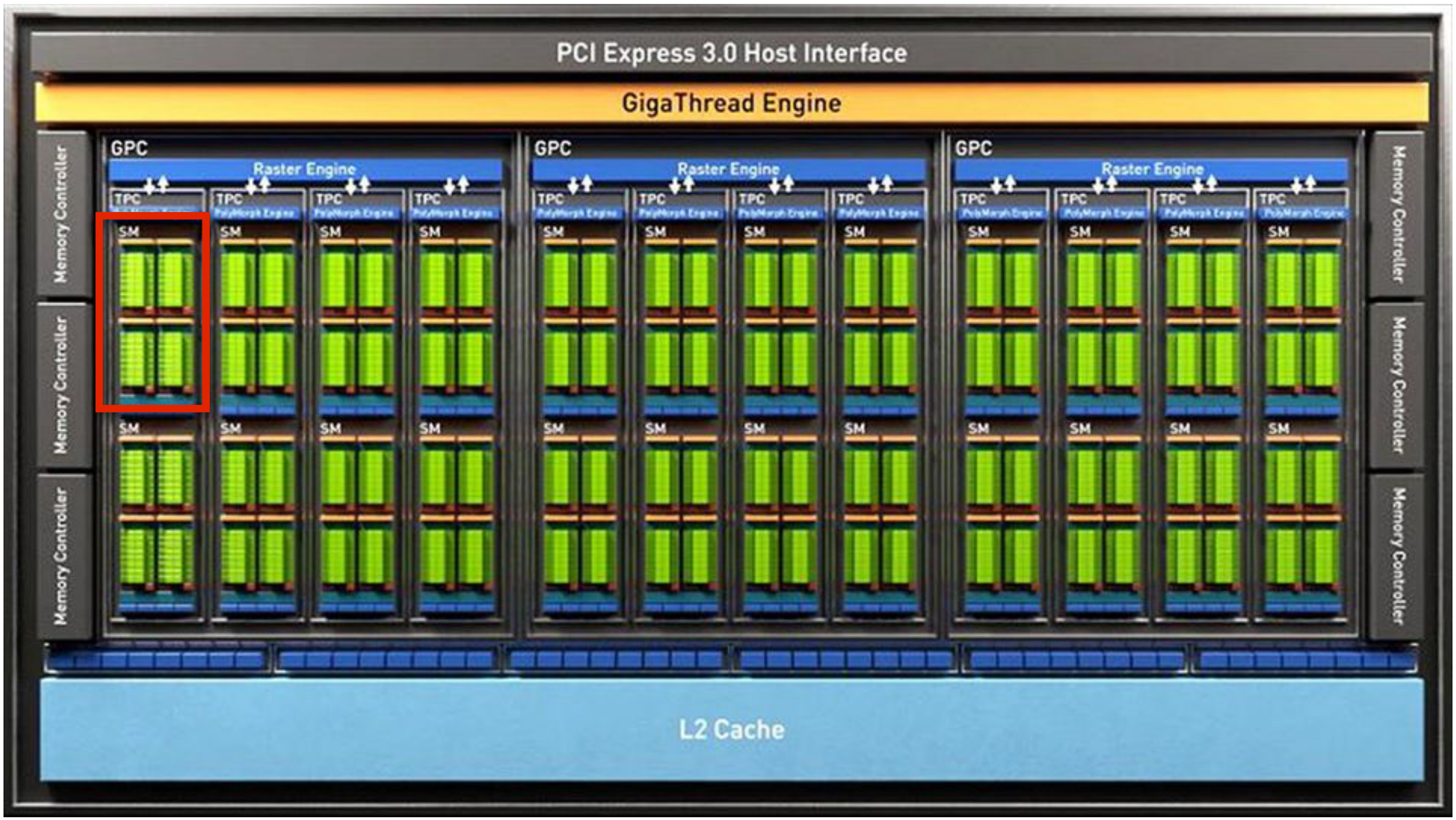


- 6 CPU Cores:
  - 2 performance (everest) and 4 efficiency (sawtooth)
- 5 GPU Cores
- Neural Engine

NVIDIA GeForce GTX 1660 Ti GPU (2019)

24 major processing blocks

(1536 “CUDA cores” … details in upcoming classes)



Supercomputing

- Today: clusters of multi-core CPUs + GPUs
- Oak Ridge Lab's Frontier (fastest supercomputer in the world)
  - CPU nodes: 9,472 AMD Epyc 7A53s "Trento" 64 core 2 GHz
  - GPU nodes: 37,888 Radeon Instinct MI250X GPUs
  - Grand total: 606,208 CPU cores + 8,335,360 GPU cores

What is a parallel computer?

One common definition

A parallel computer is a **collection of processing elements** that **cooperate** to **solve problems** **quickly**

We care about performance *
We care about efficiency

We're going to use multiple processors to get it

* Note: different motivation from "concurrent programming" using pthreads in 15-213

DEMO 1

(This semester's first parallel program)

Speedup

One major motivation of using parallel processing: achieve a speedup

For a given problem:

$$\text{speedup(using P processors)} = \frac{\text{execution time (using 1 processor)}}{\text{execution time (using P processors)}}$$

Class observations from demo 1

- **Communication limited the maximum speedup achieved**
  - In the demo, the communication was telling each other the partial sums

- **Minimizing the cost of communication improves speedup**
  - Moving students ("processors") closer together (or let them shout)

DEMO 2

(scaling up to four “processors”)

Class observations from demo 2

- **Imbalance in work assignment limited speedup**
  - Some students ("processors") ran out work to do (went idle), while others were still working on their assigned task

- **Improving the distribution of work improved speedup**

DEMO 3

(massively parallel execution)

Class observations from demo 3

- The problem I just gave you has a significant amount of communication compared to computation

- Communication costs can dominate a parallel computation, <u>severely limiting</u> speedup

Course theme 1:

Designing and writing parallel programs ... that scale!

- **Parallel thinking**
  1. Decomposing work into pieces that can safely be performed in parallel
  2. Assigning work to processors
  3. Managing communication/synchronization between the processors so that it does not limit speedup

- **Abstractions/mechanisms for performing the above tasks**
  - Writing code in popular parallel programming languages

Course theme 2:

Parallel computer hardware implementation: how parallel computers work

- Mechanisms used to implement abstractions efficiently
  - Performance characteristics of implementations
  - Design trade-offs: performance vs. convenience vs. cost

- Why do I need to know about hardware?
  - Because the characteristics of the machine really matter (recall speed of communication issues in earlier demos)
  - Because you care about efficiency and performance (you are writing parallel programs after all!)

Course theme 3:
Thinking about efficiency

- **FAST != EFFICIENT**

- **Just because your program runs faster on a parallel computer, it does not mean it is using the hardware efficiently**
  - Is 2x speedup on computer with 10 processors a good result?

- **Programmer's perspective: make use of provided machine capabilities**

- **HW designer's perspective: choosing the right capabilities to put in system (performance/cost, cost = silicon area?, power?, etc.)**

Fundamental Shift in CPU Design Philosophy

Before 2004:

- within the chip area budget, maximize performance
  - increasingly aggressive speculative execution for ILP

After 2004:

- area *within* the chip matters (limits # of cores/chip):
  - maximize performance per area
- power consumption is critical (battery life, data centers)
  - maximize performance per Watt
- upshot: major focus on *efficiency* of cores

Summary

- Today, single-thread performance is improving very slowly
  - To run programs significantly faster, programs must utilize multiple processing elements
  - Which means you need to know how to write parallel code
- Writing parallel programs can be challenging
  - Requires problem partitioning, communication, synchronization
  - Knowledge of machine characteristics is important
- I suspect you will find that modern computers have tremendously more processing power than you might realize, if you just use it!
- Welcome to 15-418!