CSE 6220/CX 4220
Introduction to HPC

Lecture 1: Course Logistics & Introduction to HPC

Helen Xu

Georgia Tech College of Computing
School of Computational Science and Engineering

(Some slides from Srinivas Aluru's/ Ümit V. Çatalyürek's CSE 6220/CS4220,
Rich Vuduc's/Ramki Kannan's CSE 6230, MIT's OCW 6.172,
UC Berkeley CS267, Utah's CS 6530)

Course policy on mobile devices

no smartphones

no laptop

or other mobile devices e.g., tablet

Why? Research shows that they slow us down in a learning environment.

(I will post all slides and lecture videos online.)

Inspired by: https://users.cs.utah.edu/~pandey/courses/cs6530/fall23/index.html

Ask questions

…and answer my questions.

Our goal in this class is to have **interesting discussions** that will help to understand the material.

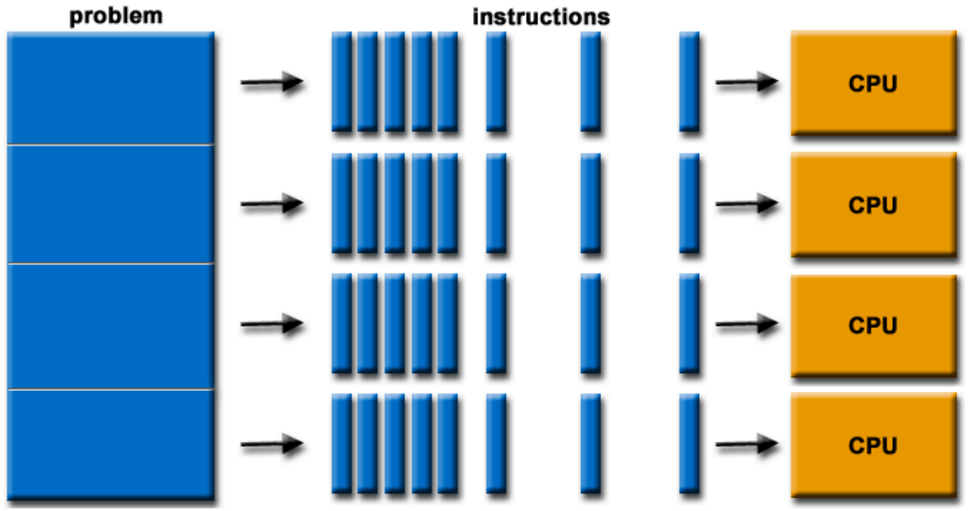
Please speak up if anything is unclear / with any questions / comments!

Inspired by: https://users.cs.utah.edu/~pandey/courses/cs6530/fall23/index.html

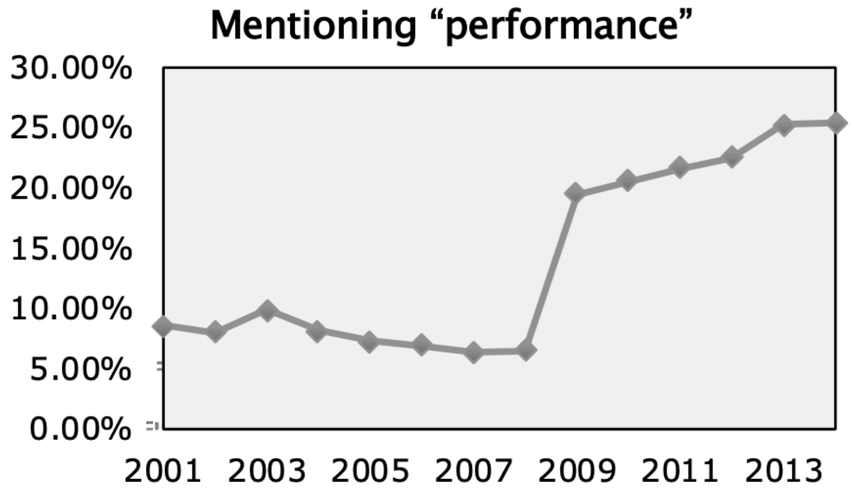
Why should you take this course?

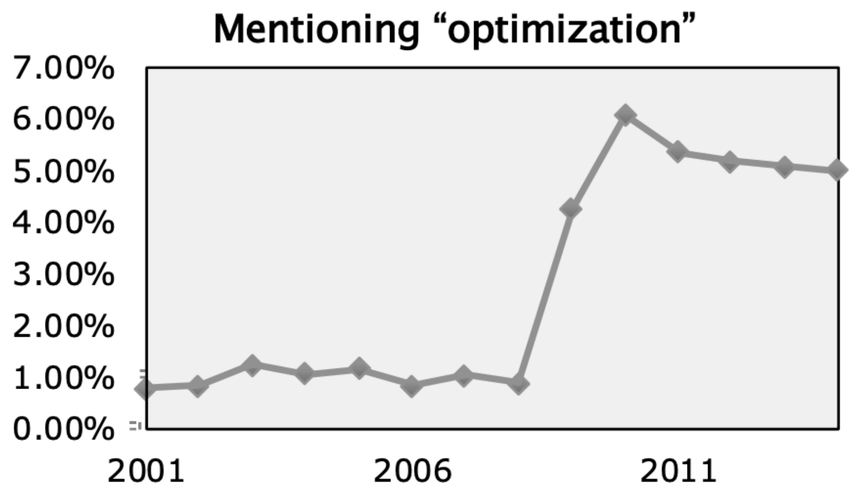
Almost all **modern computers are parallel**. The algorithms and code we develop need to run efficiently on these machines.

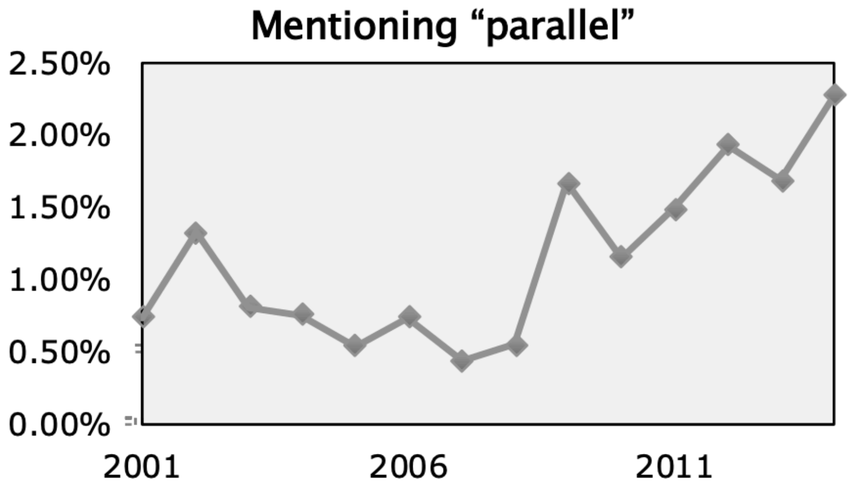
Performance engineers are in demand and will be necessary as problem and data sizes continue to grow.

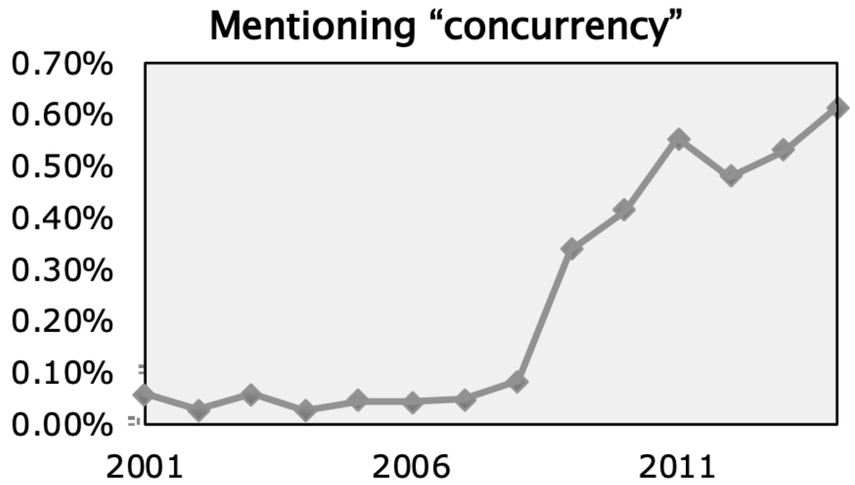


Software developer jobs









Source: Monster.com

Course objectives

At the end of the course, students should be able to:

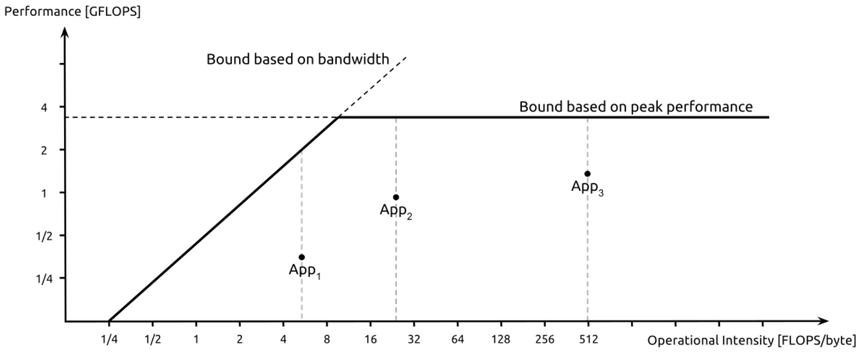
- Design and analyze parallel algorithms,
- Build advanced parallel algorithms using parallel primitives, and
- Implement parallel programs using MPI.

Theory  Practice

What we will cover

primarily theory, some implementation

- **Memory hierarchy** and analysis models
- **Parallel algorithm analysis**: Speedup, efficiency, and scalability
- **Parallel Algorithms primitives**: Prefix sums and applications, communication primitives, sorting, matrix algorithms, Fast Fourier transforms, graph algorithms
- **Models**: Communication networks, embeddings
- **General Techniques**: Decomposition techniques, load balancing, mapping
- **Parallel Programming with MPI**: Writing and executing MPI programs, collective communication, grouping data for communication, communicators and topologies



What we will not cover

Introductory algorithms and data structures

If you need a review on algorithm (e.g., big-O) analysis, see **CLRS** (posted on Canvas)

See: Introduction to Algorithms (CLRS)

Asymptotic cheat sheet

Concurrent algorithms and data structures

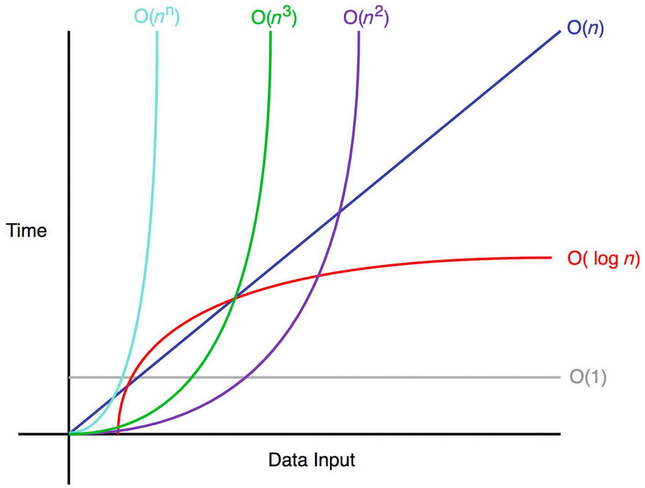
See CSE 6230 from last semester, or Guy Blelloch's "Parallel and Concurrent Algorithms"

Advanced practical performance engineering

Prerequisites / Requirements

Undergraduate algorithms/data structures, ability to program in C/C++

- TA Cluster and C/C++ review via Zoom on **Friday, August 23, at 1pm.** The link will be sent out via Canvas.



+

No textbook purchase needed

- All lecture notes / materials will be posted on Canvas

Time requirements

- Depends on each student's background and experience

Course logistics

Course materials: on Canvas

Discussion board: on Ed Discussion

6220 vs. 4220: separate grading, some exam problems may be different

Academic honesty:

- Refer to the GT Academic Honor Code.
- If you are not sure, ask me.
- Seriously, do **not** plagiarize.

Inspired by: https://users.cs.utah.edu/~pandey/courses/cs6530/fall23/index.html

What is plagiarism?

Examples include:

- Listening while someone **dictates** a solution.
- Basing your solution on an **existing written solution**.
- **Copying** another student's code or **sharing** your code with other students.

Inspired by: https://users.cs.utah.edu/~pandey/courses/cs6530/fall23/index.html

What is collaboration?

Asking questions on the discussion board.

Working together to find a good approach for solving a problem.

A **high-level discussion** of solution strategy.

If you collaborate with other students, **declare it upfront** (at the start of the report).

Inspired by: https://users.cs.utah.edu/~pandey/courses/cs6530/fall23/index.html

Academic honesty

You are allowed to discuss assignments and collaborate with other students in the class. However,

- each student must write the solution on her/his own and
- explicitly mention the names of collaborating student(s) for each problem in every copy of the submission.

The following are not allowed:

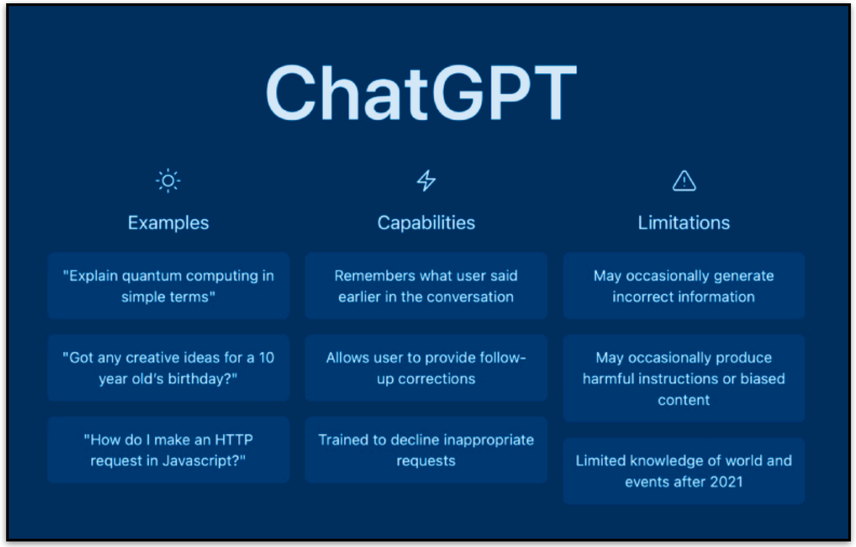
- Any reference to assignments from previous terms or web postings
- Any copying of non-trivial code
  - Non-trivial = more than a line or so
  - Includes reading someone else's code and then going off to write your own.

Usage of AI assistants

In this course, we will treat **AI-based assistance** (e.g., ChatGPT and Github Copilot) the same way we treat **collaboration with other people**.

For AI tools, we recommend the following heuristics:

- **Do not copy** from the AI assistant into your assignments.
- **Do not have** your assignment and the AI assistant **open at the same time**.



From https://www.cc.gatech.edu/news/new-policies-navigate-role-ai-assistants-cs-courses

Course team

Prof. Helen Xu
Email:
hxu615@gatech.edu
Office hours: via Zoom,
11a-12pm Tuesdays

Abhishek Kalokhe
Office hours: 11a-12p Mondays

Atharva Bhalerao
Office hours: 11a-12p Thursdays

Grade breakdown

Assignment	% worth
Academic integrity homework	1
Written homeworks	7 @ 2 each = 14
Programming assignments	1 @ 3 + 2 @ 6 = 15
Exam 1	20
Exam 2	20
Final exam	30
Total	100

Assignment deadlines

Assignments are due **11:59pm on the due date** of the assignment.

Every assignment (besides exams) have a **free 48 hour grace period**. No assignments will be accepted after the grace period without a documented emergency. You do not need to contact any of the staff if you are using the extra time.

Grading

Grading follows the standard scheme:

- A = [90, 100]
- B = [80, 90)
- C = [70, 80)
- D = [60, 70)
- F = [0, 60)

Pass/Fail Policy

- Do all homeworks, programming assignments, and take exams.
- Earn at least 50% of the weighted total.

Grading notes

Grading style

- Learning is more important than grades
- Homework and exams can contain tough problems.
- Grading follows curve style to accommodate.

If you leave a question blank on an exam of homework, you automatically get 25% credit on that question. You may get less than 25% on the question if you plagiarize or answer completely incorrectly.

Written homeworks

There will be 7 written homeworks + 1 academic integrity homework.

The written homeworks can be done in groups if you write your teammates on the page. The academic integrity homework must be done individually.

All written homeworks must be typeset in LaTeX.

- We will include an example template on Canvas, or you are free to use your own.
- Here's a quick Overleaf tutorial and a more in-depth introduction.

Programming assignments

Do all development on PACE ICE (Instructional Cluster Environment)

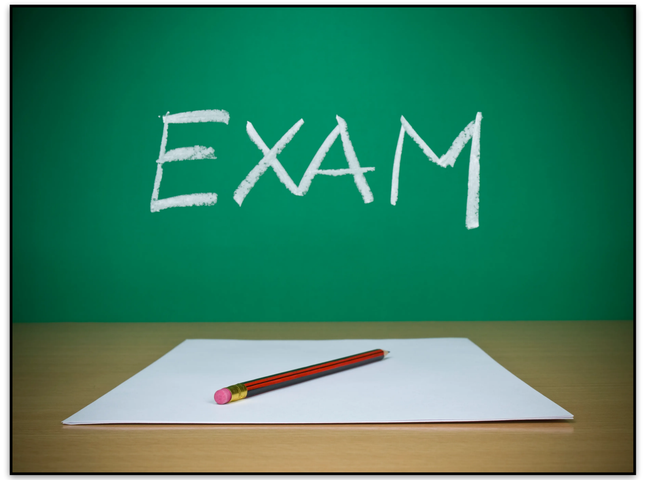
- Instructions for logging onto the cluster can be found here.

In total, there will be 3 **programming assignments** in MPI.

The assignments will be done in groups of up to 3.

The code and any other deliverables will be submitted via Canvas/Gradescope.

Exams



- Three exams total - 2 in-class midterms and one final
  - Midterm 1 - September 23
  - Midterm 2 - October 30
  - Final - Dec 6 (according to the final exam matrix)

- If you are unable to take a midterm exam at the scheduled time due to valid reasons (e.g., conference travel), please make a request at least 3 weeks prior to the midterm date and be prepared to take it earlier than the scheduled time.

- Each student will be allowed one handwritten letter-sized (8.5in x 11in) crib sheet per exam.

Accommodations

Students with disabilities

- Office of disability services: https://disabilityservices.gatech.edu/
- Reach out to me regarding any special needs

COVID/Health accommodations

- Deadline flexibility when needed for healthcare issues
- If you have a medical issue, please post privately on Edstem

Plagiarism warning

All assignments and exams must be all of **your own writing / code**.

You may **not** copy source code from other groups or the web.

Plagiarism will **not** be tolerated. Students found in violation will be reported to the Office for Student Integrity (OSI).
(See the GT Academic Honor Code for additional information)

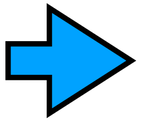
Questions?

For technical questions about the homework/exams, please use the discussion board. Do not email me or the TAs directly.

If you have an emergency, please post privately on the discussion board so that it reaches the whole course staff.

Today's Agenda


Course Logistics


Introduction to High-Performance Computing

Course overview

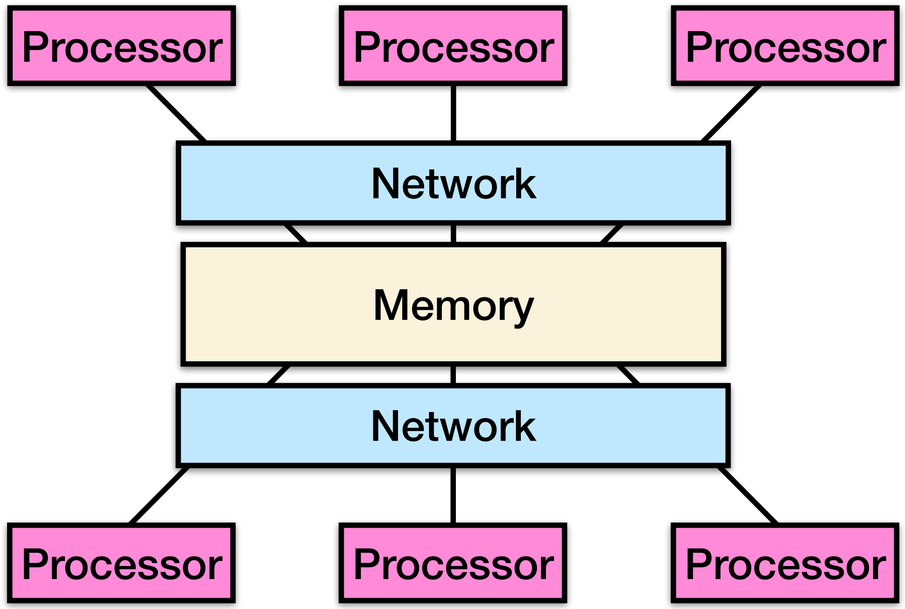
What? (Goal of the class)

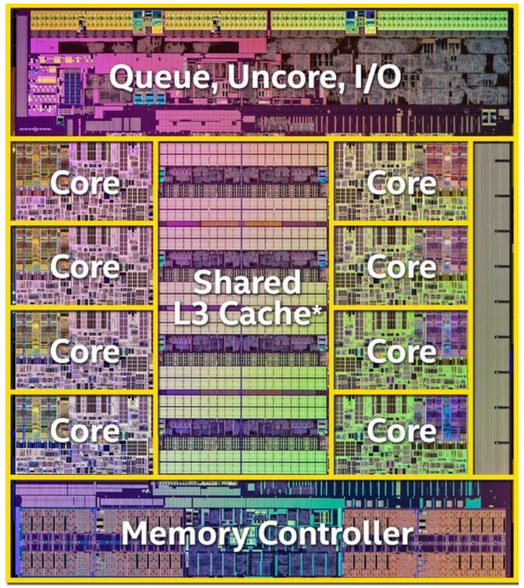
Learn **parallel computing** to take advantage of modern hardware.

Why?

- Solve **BIG** problems
  - N-body simulations, molecular dynamics, multi-scale modeling, training LLMs …
- **Time**: Improve quality of answer given limits of human patience
  - climate predictions
- **Space**: Solve memory-intensive problems
  - Big data analytics, network science, astrophysics, biology, ...

What is a parallel computer?



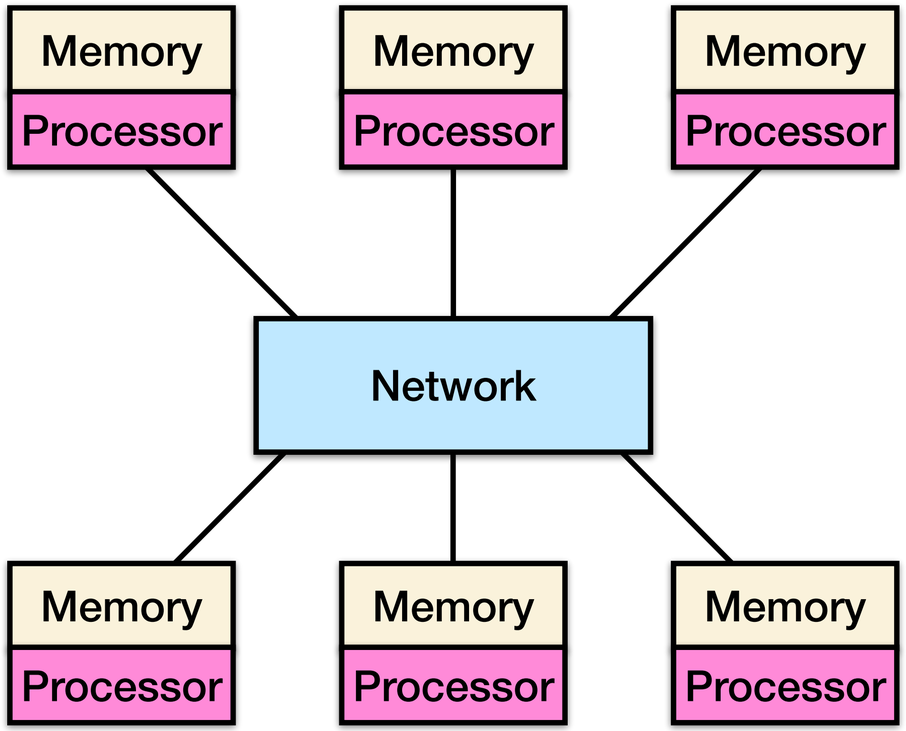


E.g. Intel Haswell

A **shared-memory multiprocessor** (SMP) connects multiple processors to a single memory system.

A multicore processor contains multiple processors (cores) on a single chip.

Scaling parallelism to multiple machines



A **distributed-memory multiprocessor** has processors with their own memories connected by a high-speed network.

Also called a **cluster**.

A **high-performance computing** (HPC) system contains 100s or 1000s of such processors (nodes).

The fastest computers (for science) have been parallel for a long time

https://www.ornl.gov/news/ornl-celebrates-launch-frontier-worlds-fastest-supercomputer

Fastest computers in the world: top500.org

Currently, the largest supercomputer (on the top 500) is Frontier (at Oak Ridge National Laboratory).

~ 8.7M cores, 22.7 kW

The first **exascale** computer - over a quintillion (a billion billion) calculations per second.

Units of measure for HPC

High-performance computing (HPC) **units** include:

- **Flop**: floating point operations (usually double precision unless otherwise noted)
- **Flop/s**: floating point operations per second
- **Bytes**: size of data (a double precision floating point number is 8 bytes)

Typical sizes are millions, billions, trillions…

Kilo	Kflop/s = 10^3 flop/s	Kbyte = $10^3 \sim 2^{10}$ bytes (KiB)
Mega	Mflop/s = 10^6 flop/s	Mbyte = $10^6 \sim 2^{20}$ bytes (MiB)
Giga	Gflop/s = 10^9 flop/s	Gbyte = $10^9 \sim 2^{30}$ bytes (GiB)
Tera	Tflop/s = 10^{12} flop/s	Tbyte = $10^{12} \sim 2^{40}$ bytes (TiB)
Peta	Pflop/s = 10^{15} flop/s	Pbyte = $10^{15} \sim 2^{50}$ bytes (PiB)
Exa	Eflop/s = 10^{18} flop/s	Ebyte = $10^{18} \sim 2^{60}$ bytes (EiB)
Zetta	Zflop/s = 10^{21} flop/s	Zbyte = $10^{21} \sim 2^{70}$ bytes (ZiB)
Yotta	Yflop/s = 10^{24} flop/s	Ybyte = $10^{24} \sim 2^{70}$ bytes (YiB)

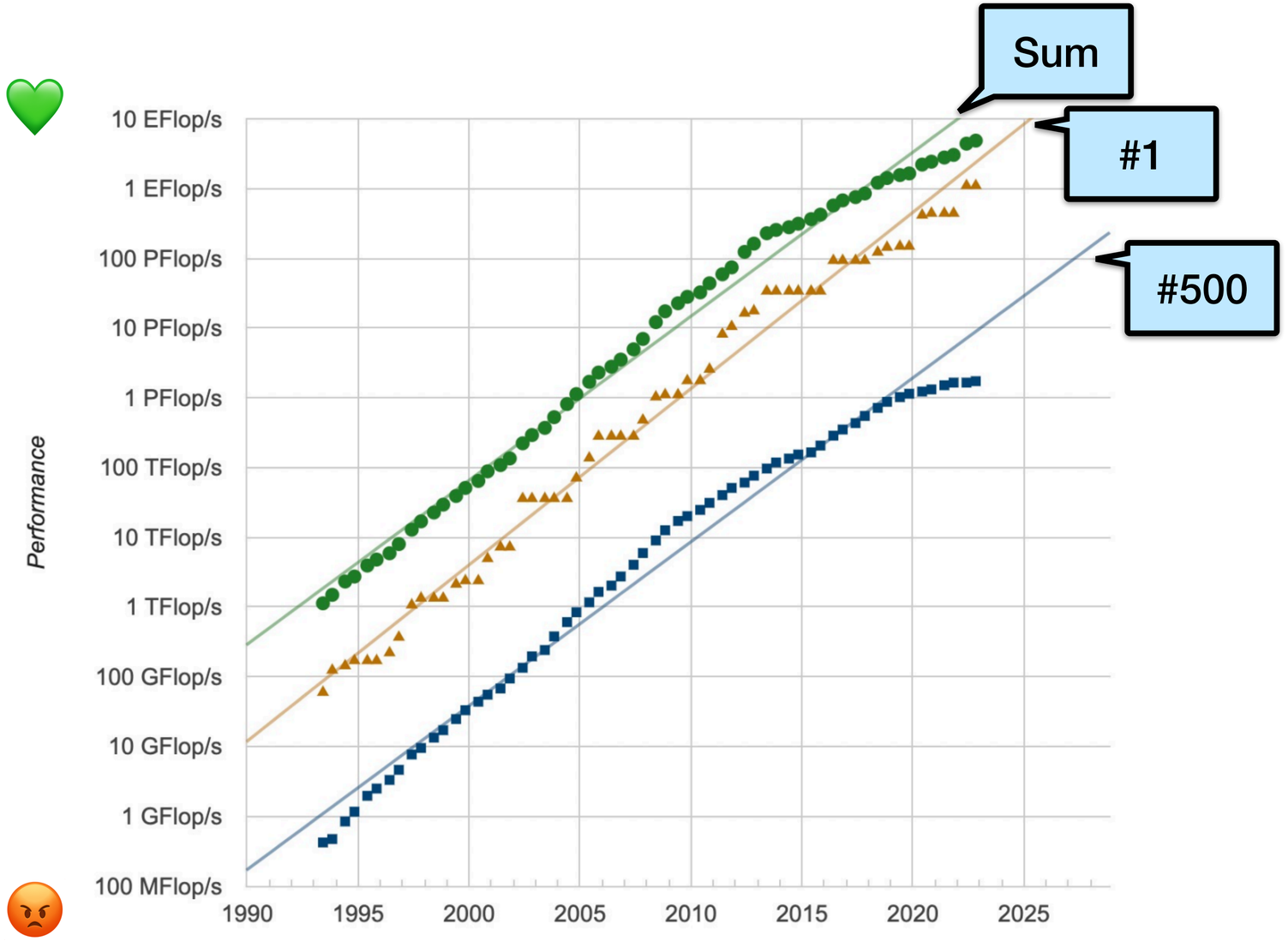
From UC Berkeley CS267

TOP 500
The List.

Nov 2023

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

Performance history and projection



Yardstick: floating point operations per second (flop/s) Rmax of **LINPACK**.

Solve **Ax=b**, Matrix A is dense with random entries.

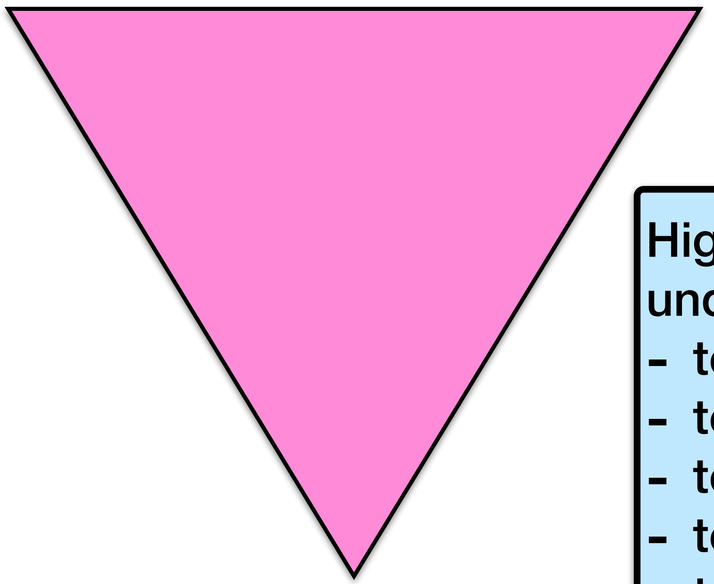
Dominated by **dense matrix-matrix multiply**.

https://www.top500.org/
(As of 1/10/22)

https://www.top500.org/project/linpack/

Science Using High-Performance Computing

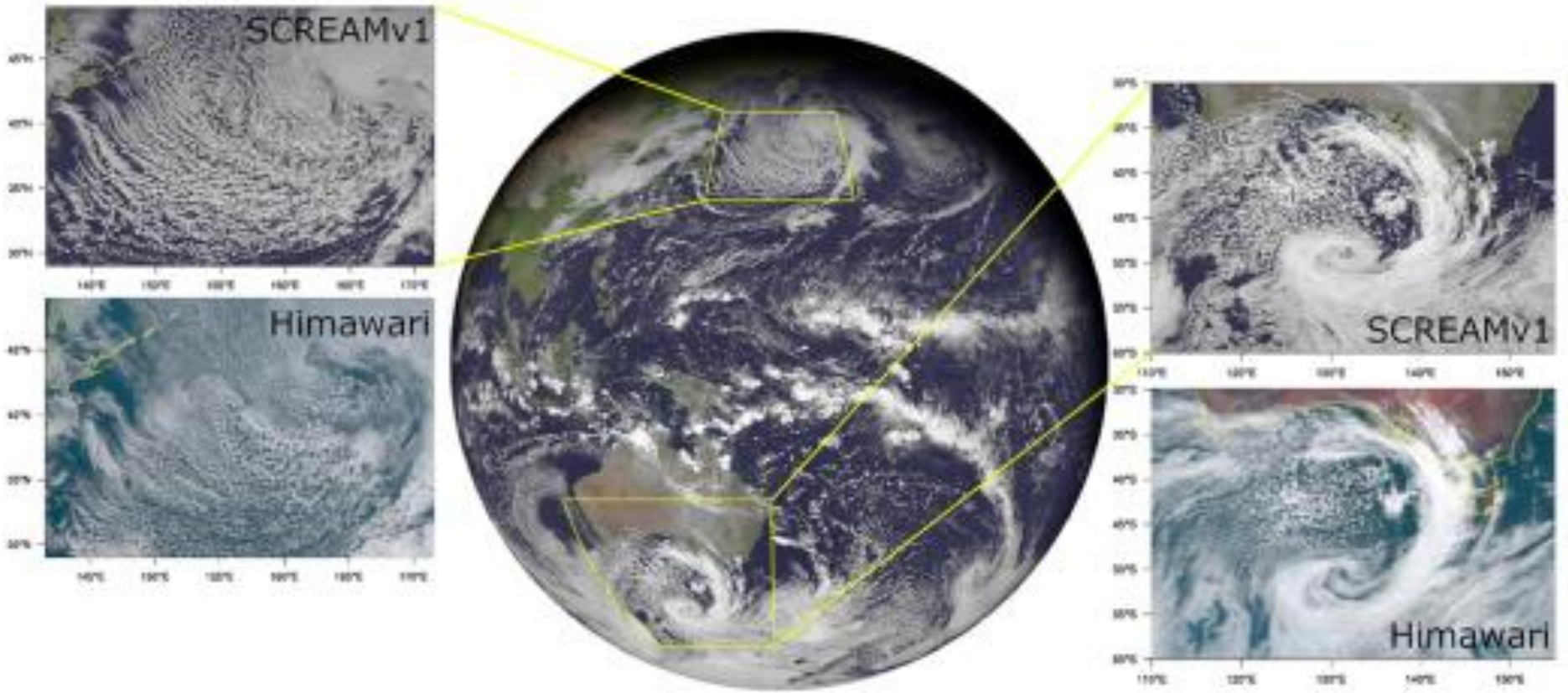
Simulation: The Third Pillar of Science



High-performance simulation is used to understand things that are:
- too big
- too small
- too fast
- too slow
- too expensive
- too dangerous

for experiments

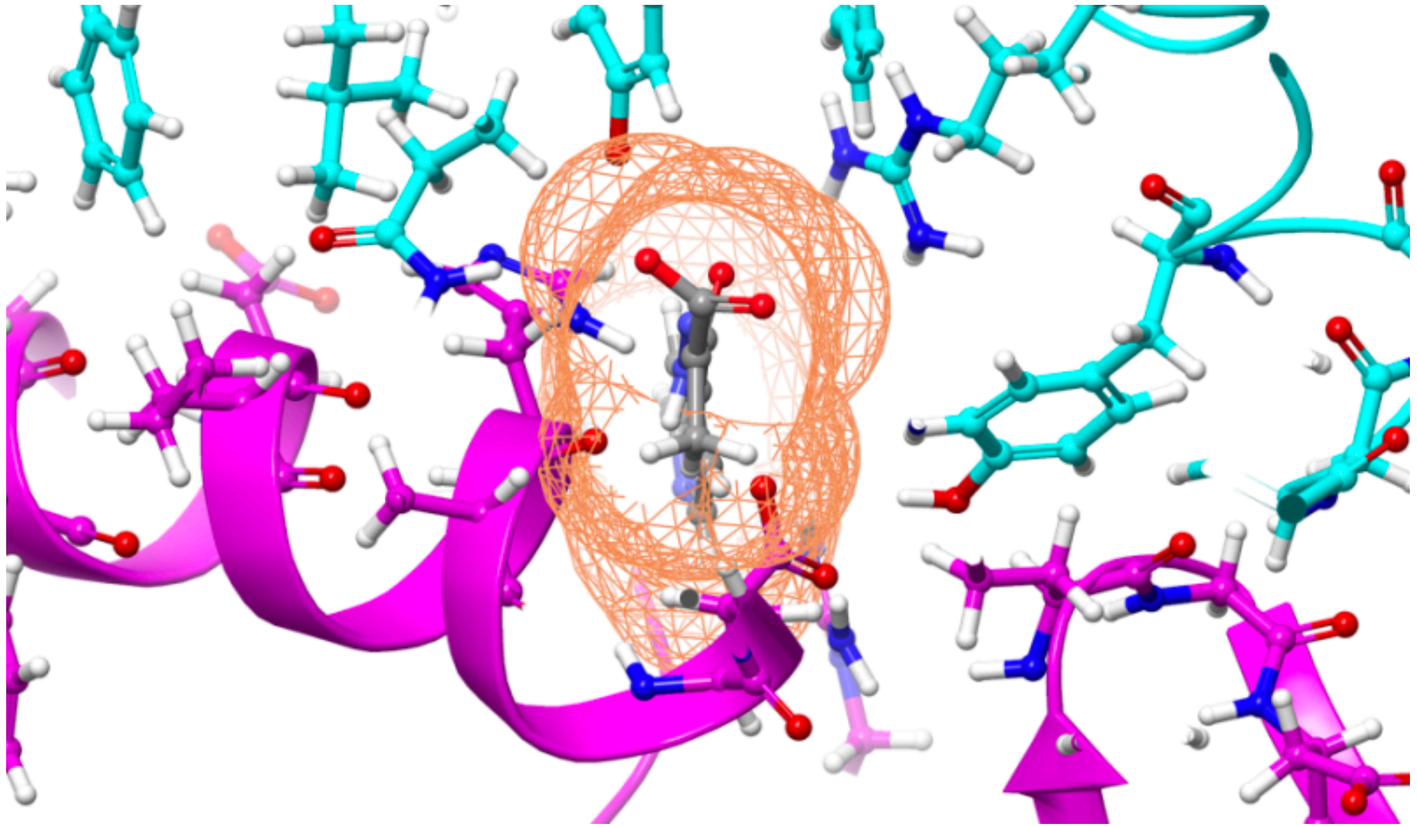
The Simple Cloud-Resolving E3SM Atmosphere Model Running on the Frontier Exascale System



https://dl.acm.org/doi/10.1145/3581784.3627044

https://climatemodeling.science.energy.gov/news/e3sm-wins-gordon-bell-prize-climate-modeling#:~:text=On%20November%2016%2C%202023%2C%20at,Atmosphere%20Model%20Running%20on%20the

Screening known drugs for COVID-19



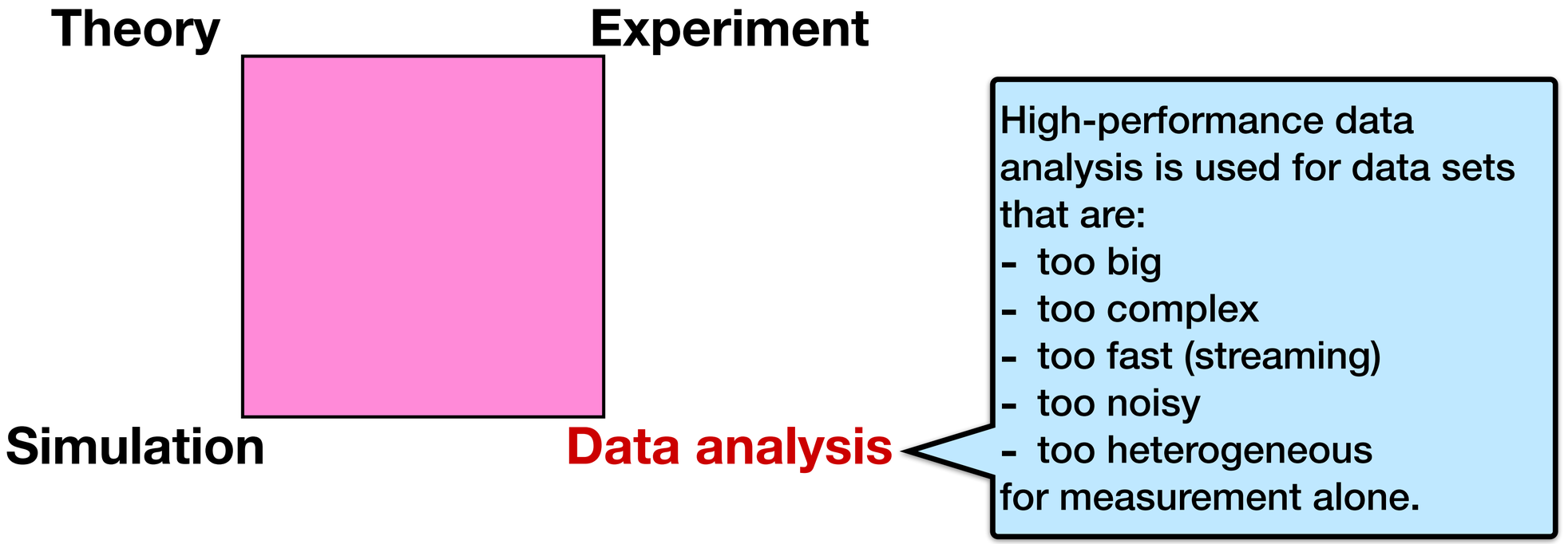
Molecular docking to SARS-Cov-2 spike protein

- Screened 8,000 compounds
- Identified 77 of the most promising

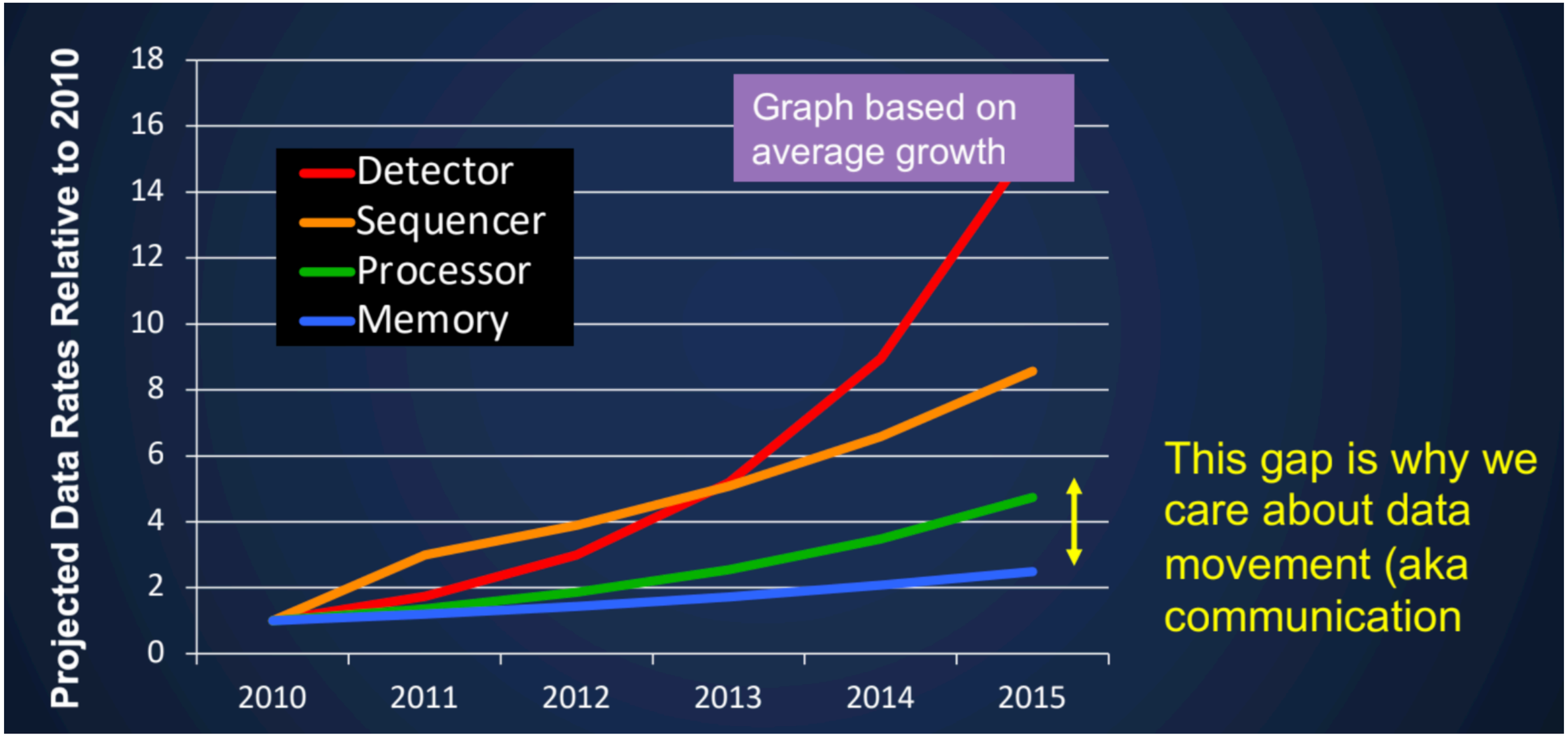
https://www.ornl.gov/news/early-research-existing-drug-compounds-supercomputing-could-combat-coronavirus

“Exascale” Scientific Applications

Data analysis: The fourth paradigm of science



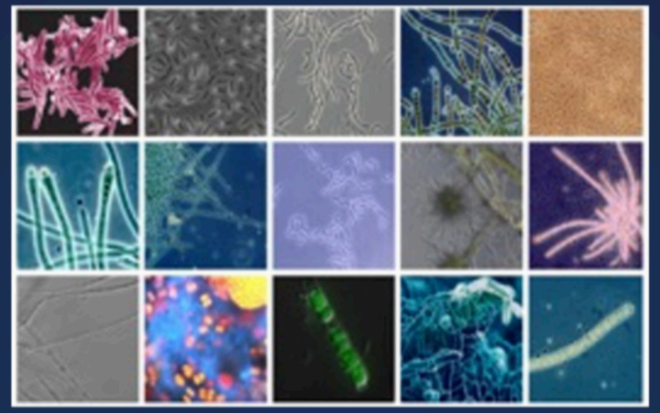
Data Growth is Outpacing Computing Growth



From UC Berkeley CS267

High Performance Data Analytics for Genomics

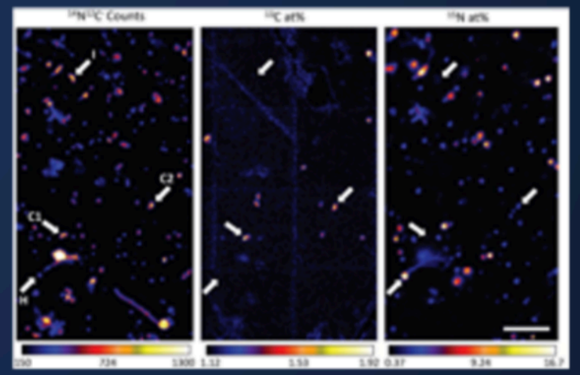
What happens to microbes after a wildfire? (1.5TB)



What are the microbial dynamics of soil carbon cycling? (3.3 TB)

How do microbes affect disease and growth of switchgrass for biofuels (4TB)

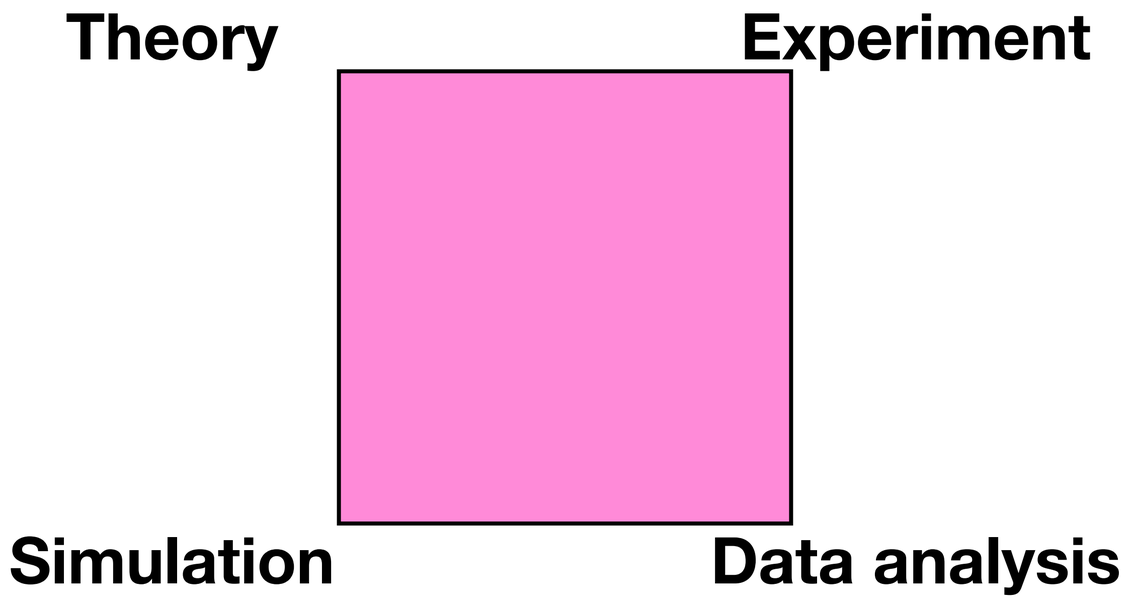
What at the seasonal fluctuations in a wetland mangrove? (1.6 TB)



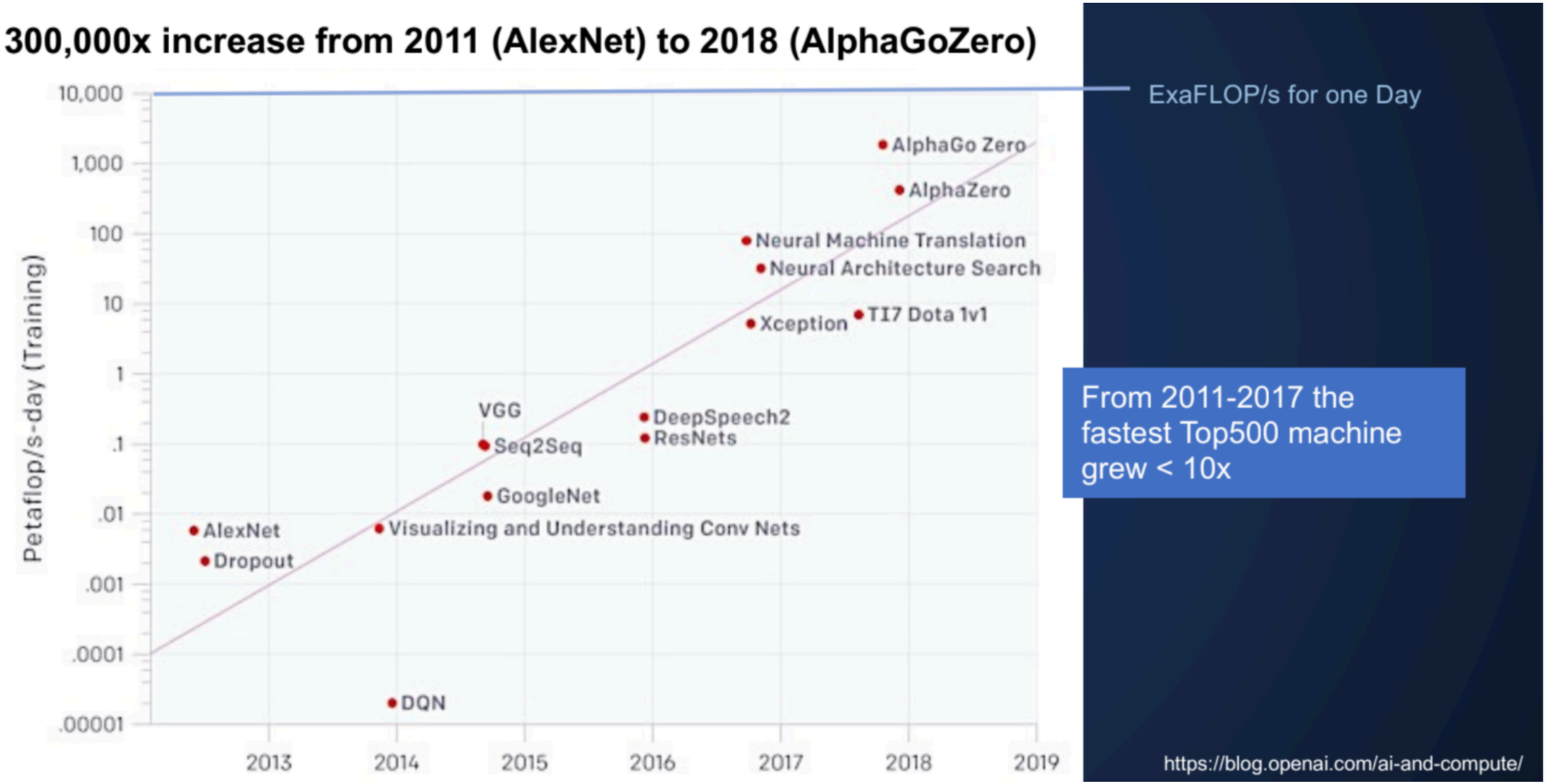
Combine genomics with isotope tracing methods for improved functional understanding (8TB)

From UC Berkeley CS267 From JGI-NERSC-KBase FICUS projects, MetaHipMer assembler ExaBiome project

The fifth paradigm of science?



Machine learning demands more computing power



From UC Berkeley CS267

Machine learning demands more computing power

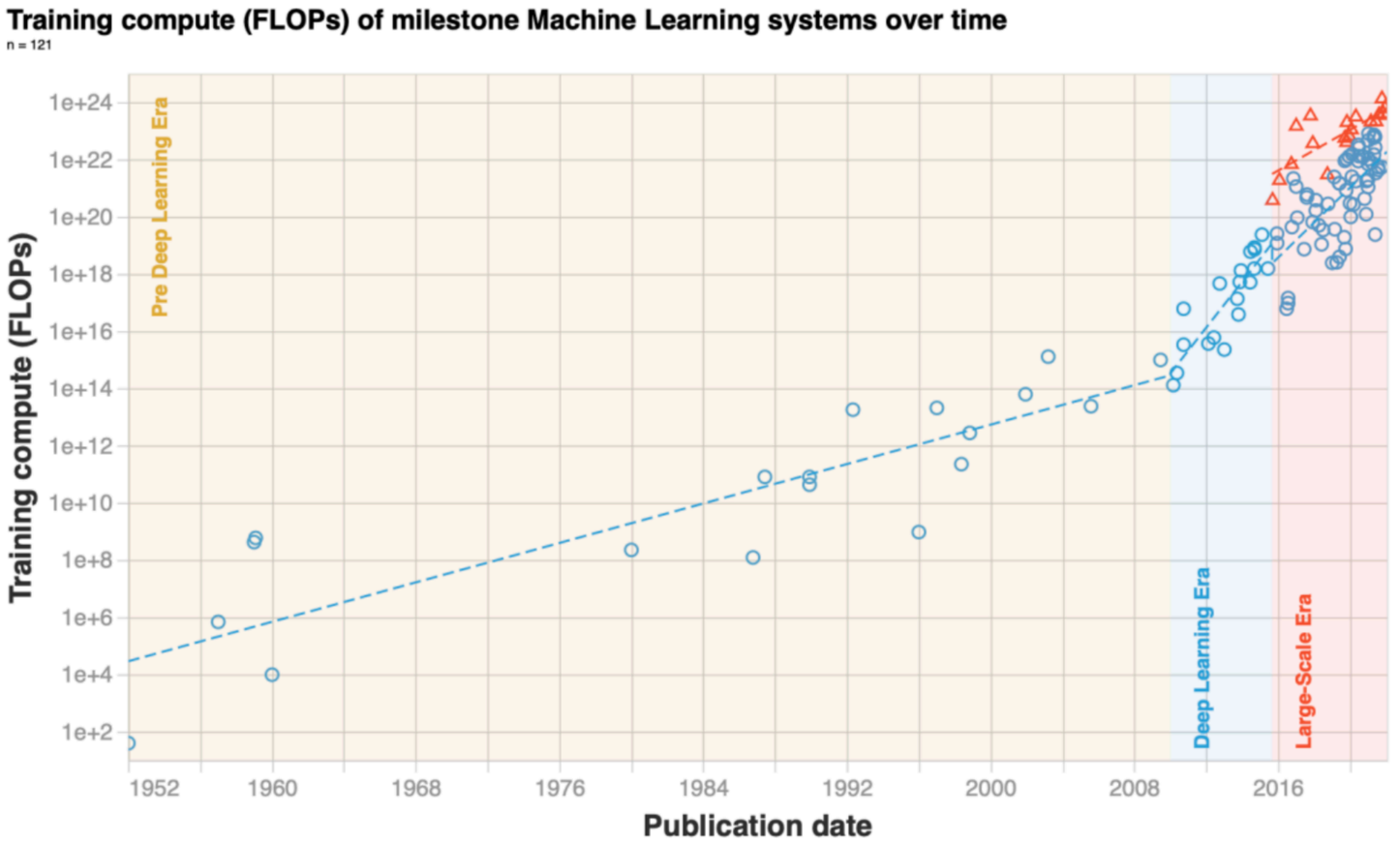


Figure 1: Trends in $n = 121$ milestone ML models between 1952 and 2022. We distinguish three eras. Notice the change of slope circa 2010, matching the advent of Deep Learning; and the emergence of a new large-scale trend in late 2015.

From UC Berkeley CS267

From "Compute trends across three eras of machine learning," Sevilla et al., 2022

AlphaFold: Predicting Protein Structure via Machine Learning

AlphaFold provides a machine-learning based solution to the grand challenge of understanding the **3D structure of proteins**.

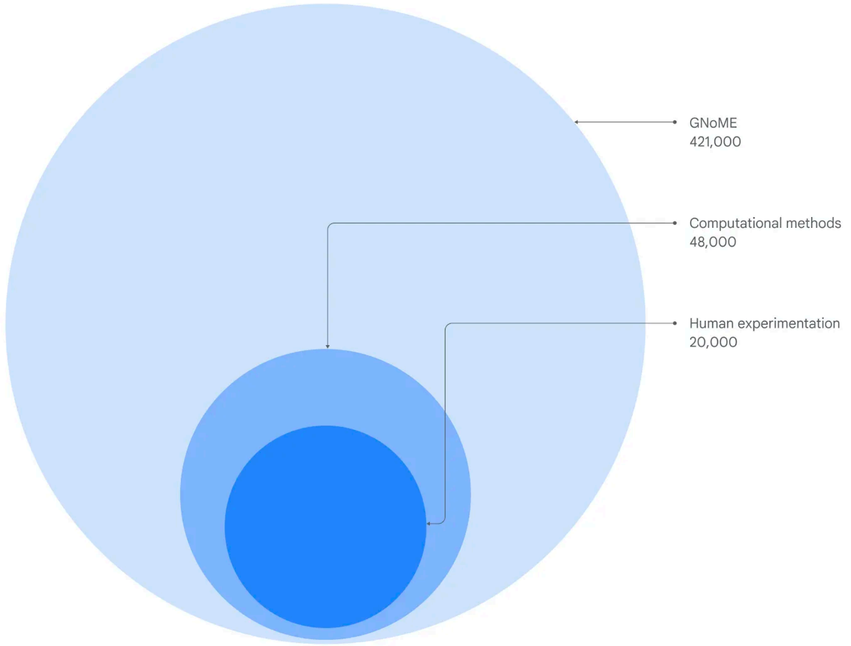
Instead of determining them experimentally (time-consuming and challenging), **AI can predict protein shape** at scale, quickly, and accurately.

https://deepmind.google/technologies/alphafold/

https://deepmind.google/discover/blog/putting-the-power-of-alphafold-into-the-worlds-hands/

Millions of new materials discovered with deep learning

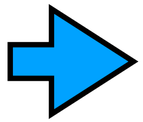
AI for materials discovery resulted in the discovery of 2.2 million new crystals - equivalent to nearly 800 years of knowledge.



https://deepmind.google/discover/blog/millions-of-new-materials-discovered-with-deep-learning/

Why are all computers nowadays parallel?

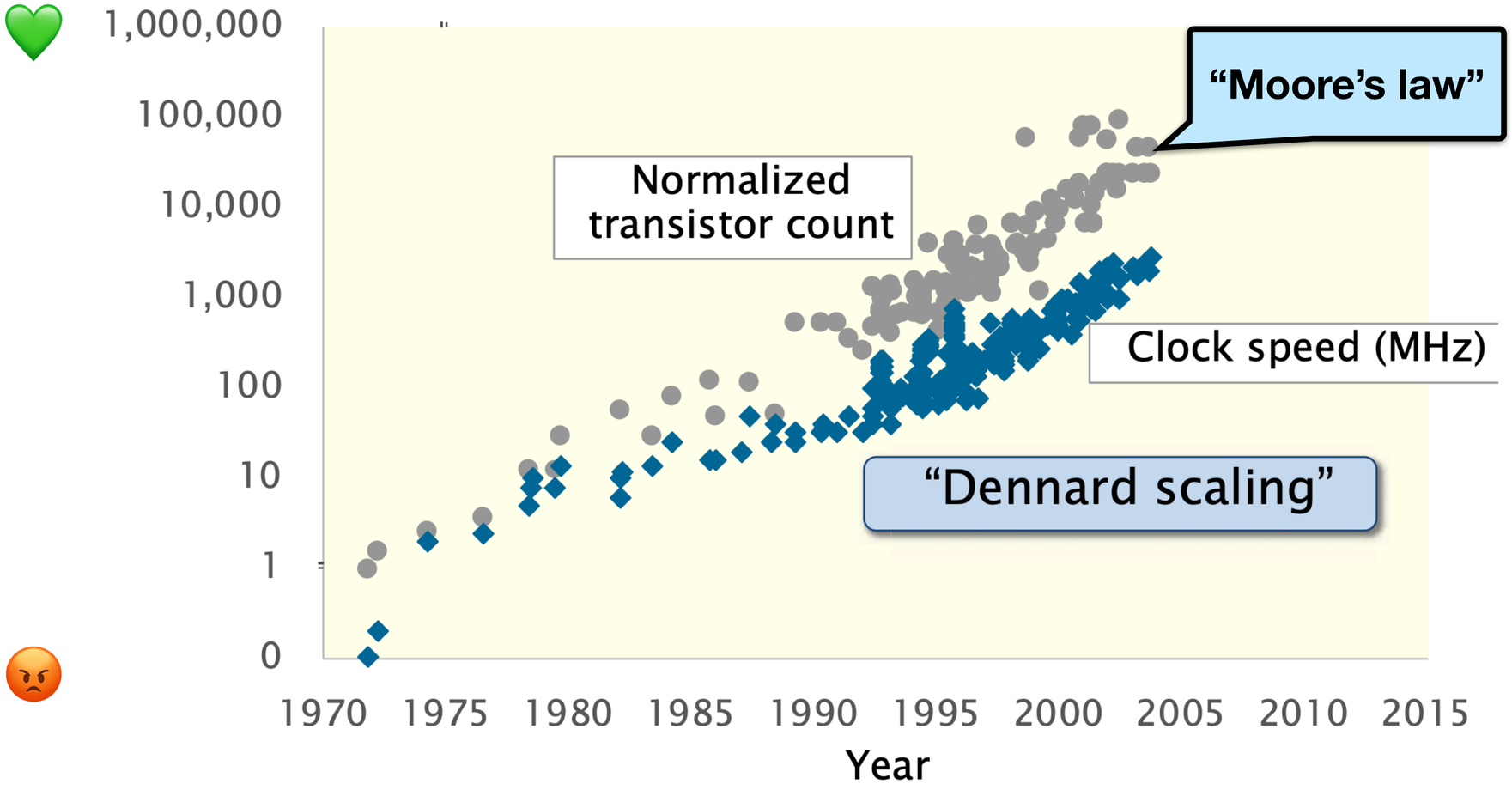
Motivating trends



How?	Driven by?
Parallelism,	**Power**,
data locality,	**memory**,
and **specialization**.	and **physics**.

From Rich Vuduc

Technology scaling before 2004



Processor data from Stanford's CPU DB [DKM12].

Until 2004, Moore's law and the scaling of clock frequency enabled **"free" performance gains** with underlying hardware changes.

Advances in hardware

Example: Apple computers with similar prices from 1977 to 2004

Courtesy of mwichary on Flickr. Used under CC-BY.

Apple II

Launched:	1977
Clock rate:	1 MHz
Data path:	8 bits
Memory:	48 KB
Cost:	$1,395

Courtesy of compudemano on Flickr. Used under CC-BY.

Power Macintosh G4

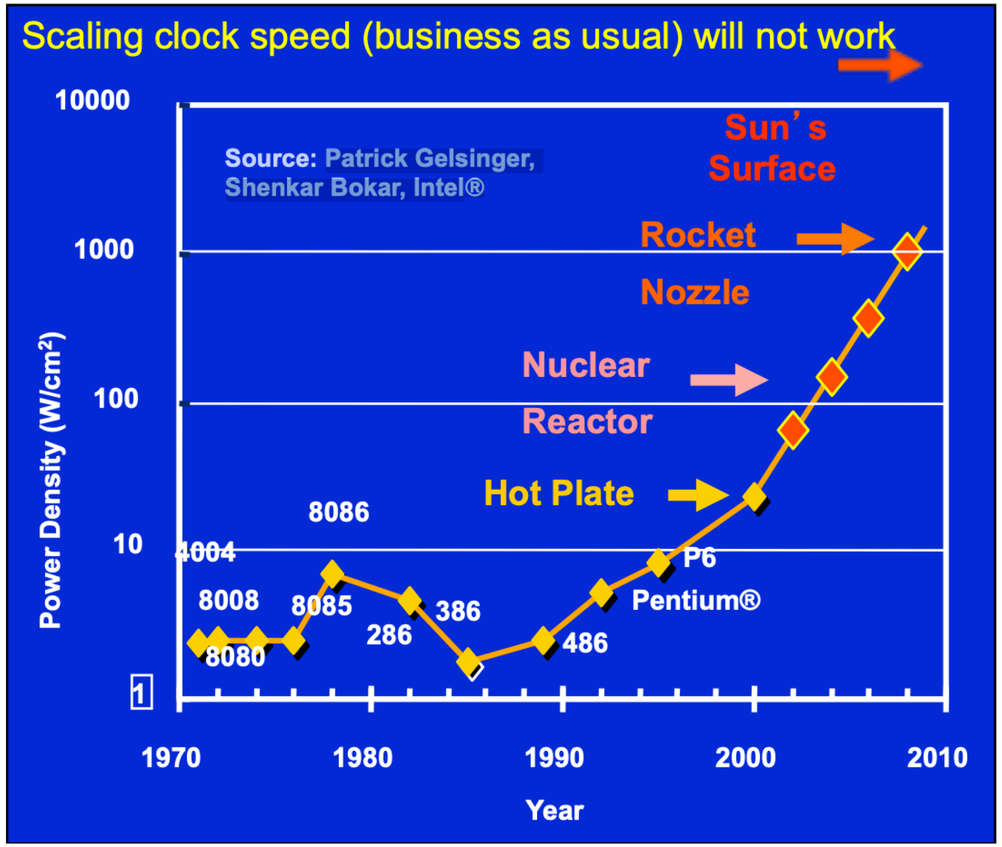
Launched:	2000
Clock rate:	400 MHz
Data path:	32 bits
Memory:	64 MB
Cost:	$1,599

Courtesy of Bernie Kohl on Wikipedia. Used under CC0.

Power Macintosh G5

Launched:	2004
Clock rate:	1.8 GHz
Data path:	64 bits
Memory:	256 MB
Cost:	$1,499

Power density



The growth of power density, as seen in 2004, if the scaling of clock frequency had continued its trend of 25%-30% increase per year.

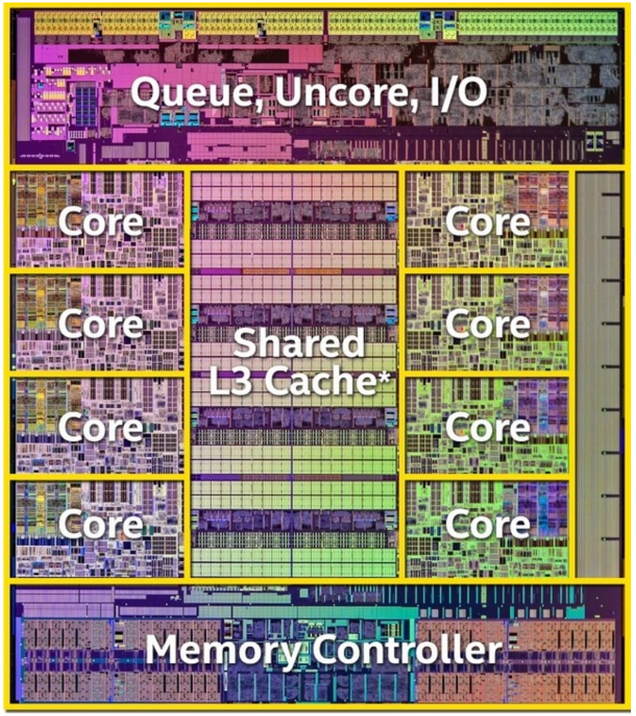
Is it better to increase speed by doubling frequency or cores?

Performance $\propto$ (cores) x (freq)

Power $\propto$ (cores) x (freq$^{2.5}$)

From Rich Vuduc

Vendor solution: Multicore (Shared-memory processors)

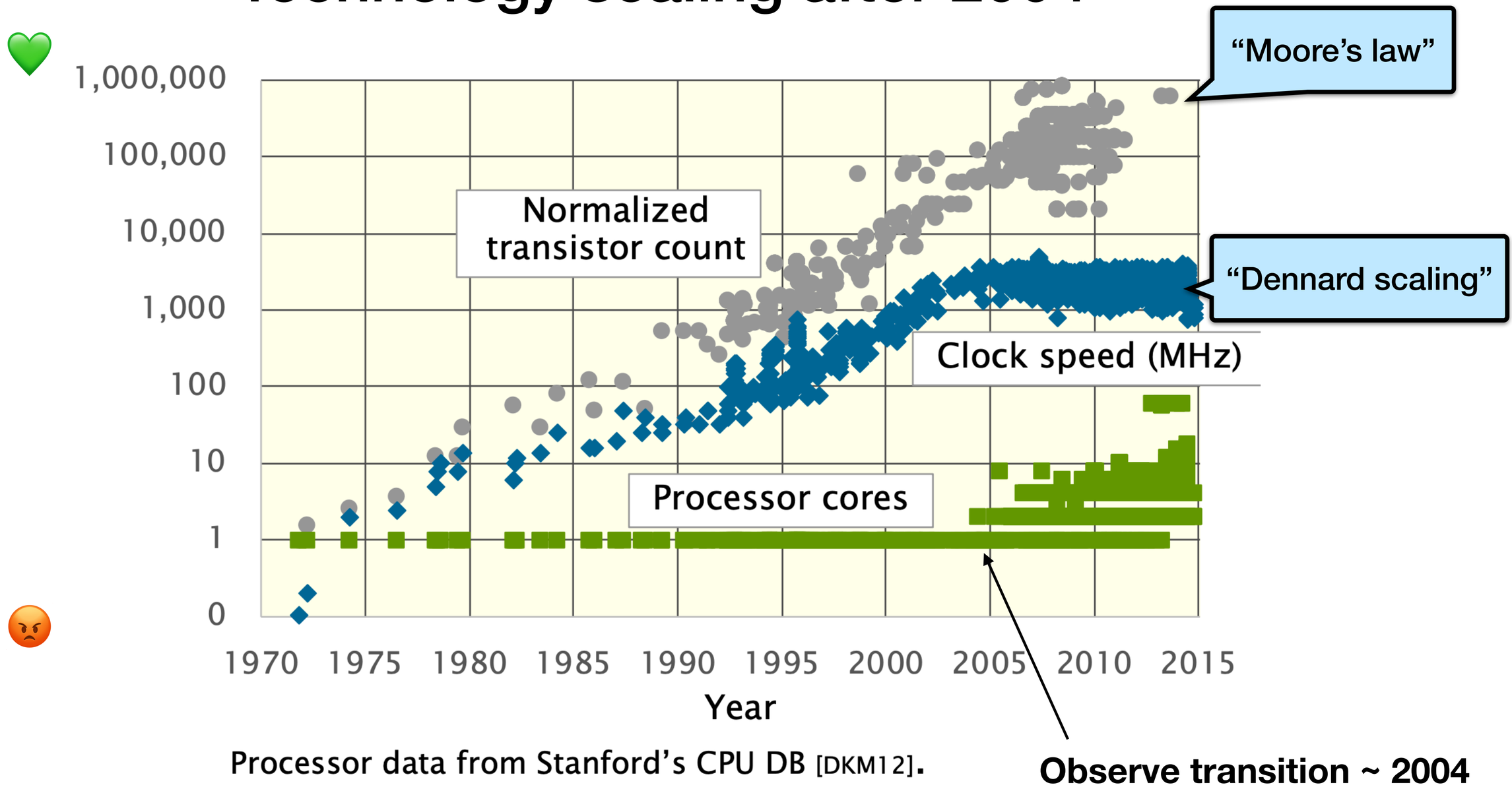


E.g. Intel Core i7 3960X
(Sandy Bridge E), 2011
6 cores
3.3 GHz
15 MB L3 cache

To scale performance, processor manufacturers put **many processing cores** on the microprocessor chip.

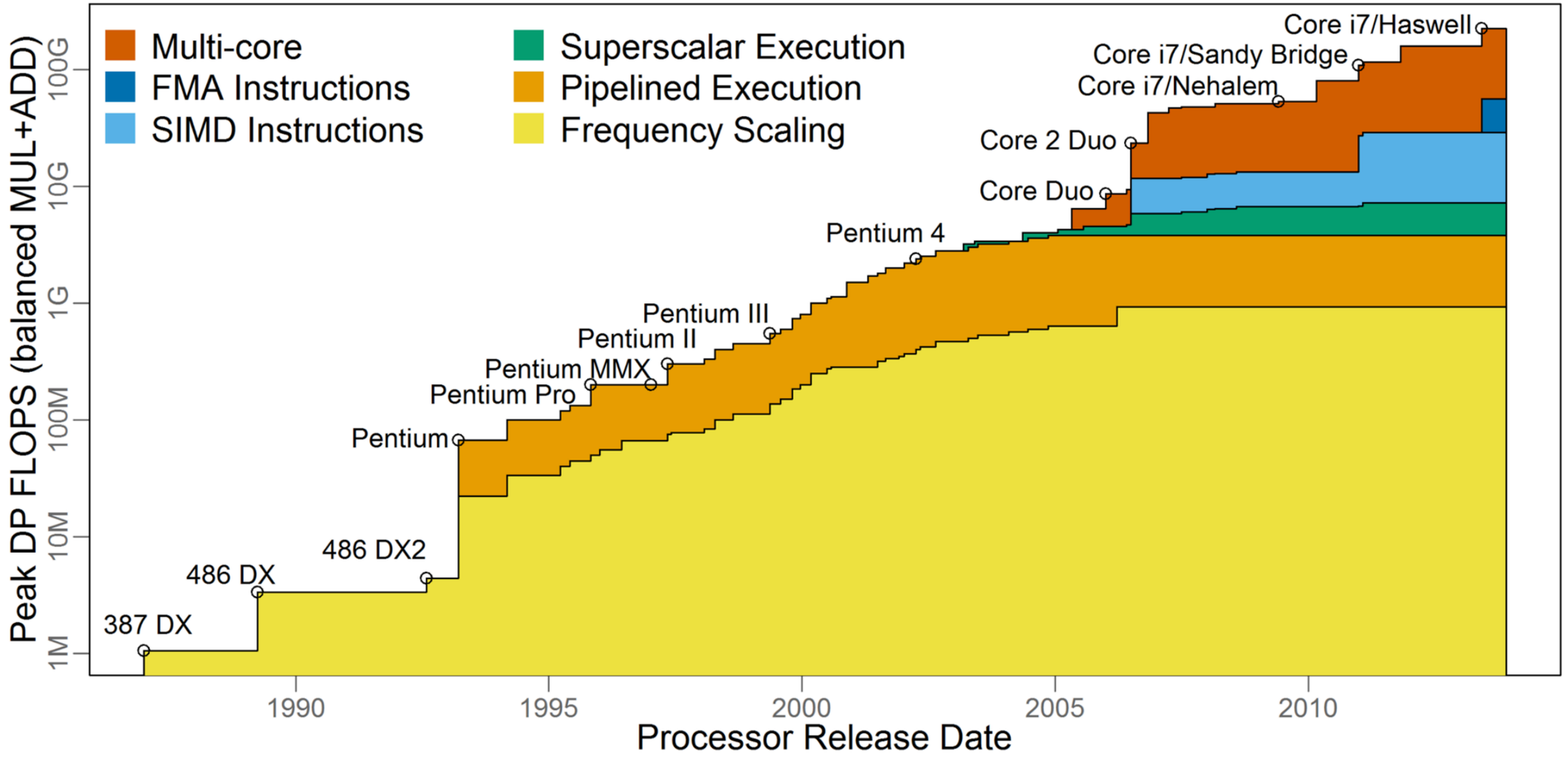
Each generation of Moore's Law potentially doubles the number of cores.

Technology scaling after 2004



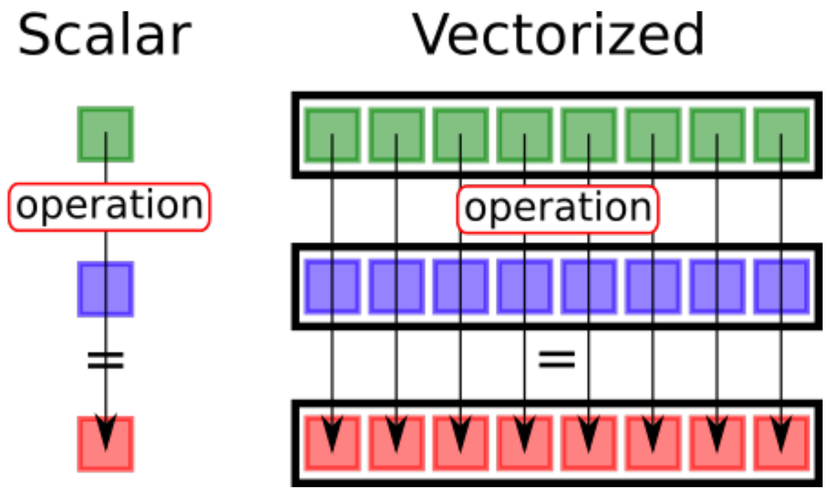
Where are performance gains coming from?

Double precision floating point ops/sec



Source: Marat Dukhan <mdukan3@gatech.edu>

Single Instruction Multiple Data (SIMD) Computer



Data-Level Parallelism
(e.g. SIMD [Flynn72]**)**

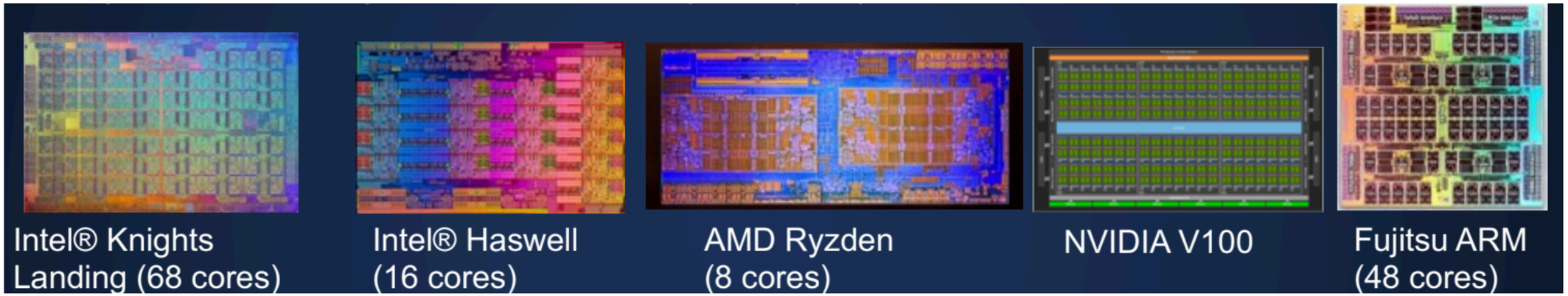
A Single-Instruction Multiple-Data (SIMD) computer has multiple processors (or functional units) that perform **the same operation on multiple data elements** at once.

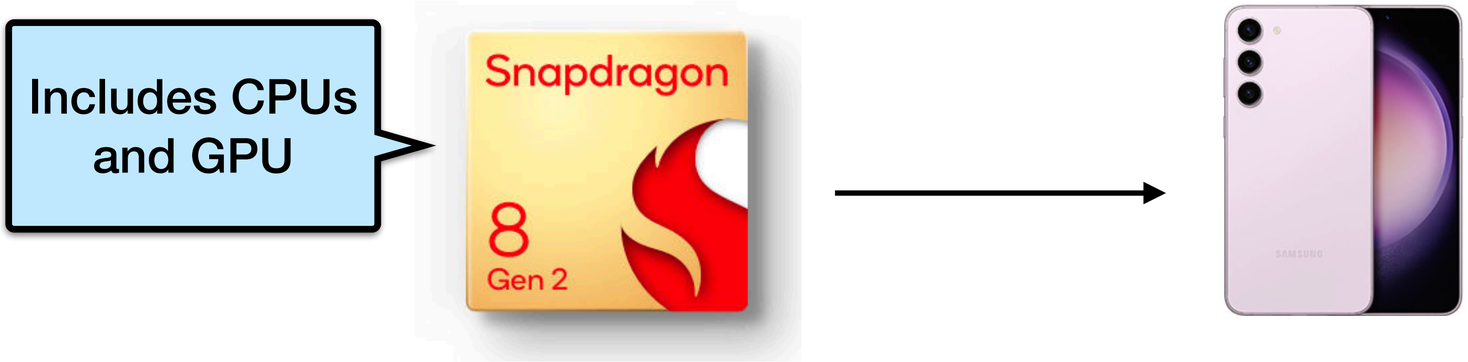
Most single processors have SIMD units with **~2-8 way parallelism**.

Graphics processing units (**GPUs**) use SIMD as their primary programming paradigm.

Parallel computing is ubiquitous in individual machines of all sizes

Trends in technology scaling drive parallelism in computing at **all scales**.

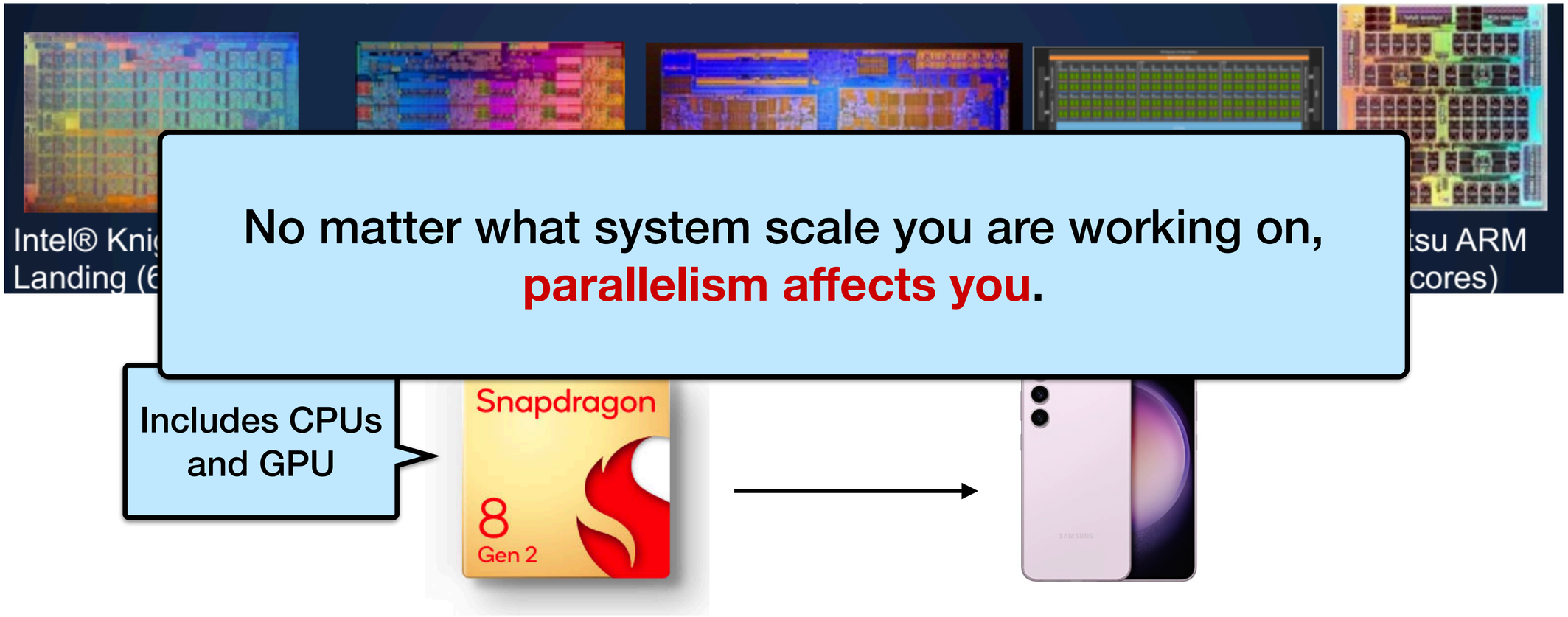




https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-8-gen-2-mobile-platform

Parallel computing is ubiquitous in individual machines of all sizes

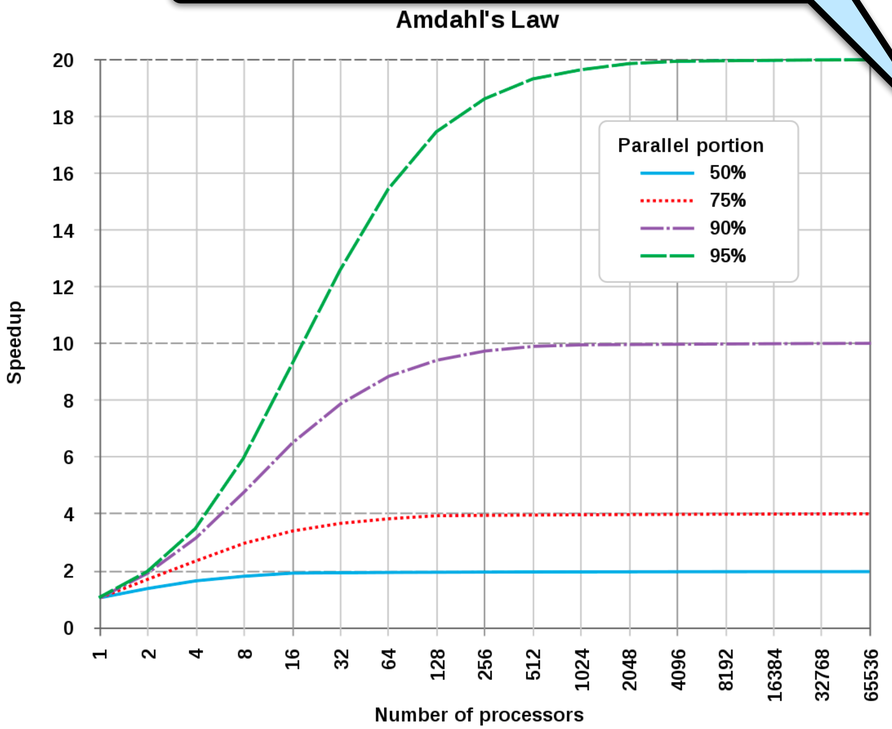
Trends in technology scaling drive parallelism in computing at **all scales**.



https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-8-gen-2-mobile-platform

Bounding Maximum Parallel Speedup with Amdahl's Law

Work done by the **best** sequential algorithm



Suppose only part of an application is parallel.

Amdahl's law:

s = fraction of work done seqentially (Amdahl fraction)

1-s = parallelizable fraction

P = number of processors

Speedup (P) = Time(1) / Time(P)

<= 1/(s + (1-s)/P)

<= 1/s

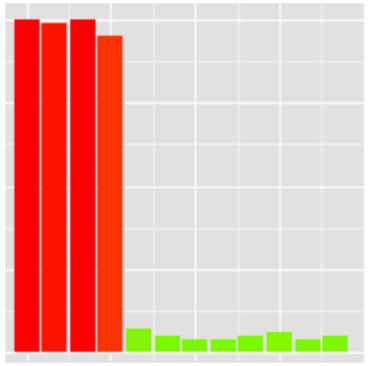
Even if the parallel part speeds up perfectly, **performance is limited by the sequential part**.

From Ramki Kannan

Writing fast code requires going beyond parallelism alone

This class is about more than just parallelism. It's about getting the most out of your hardware, e.g., through **data locality.**

Why? The faster you run, the sooner you can stop, saving time, energy, cost, …

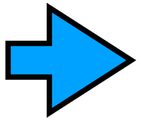


"Race-to-halt" — finish quickly then shutdown to save energy

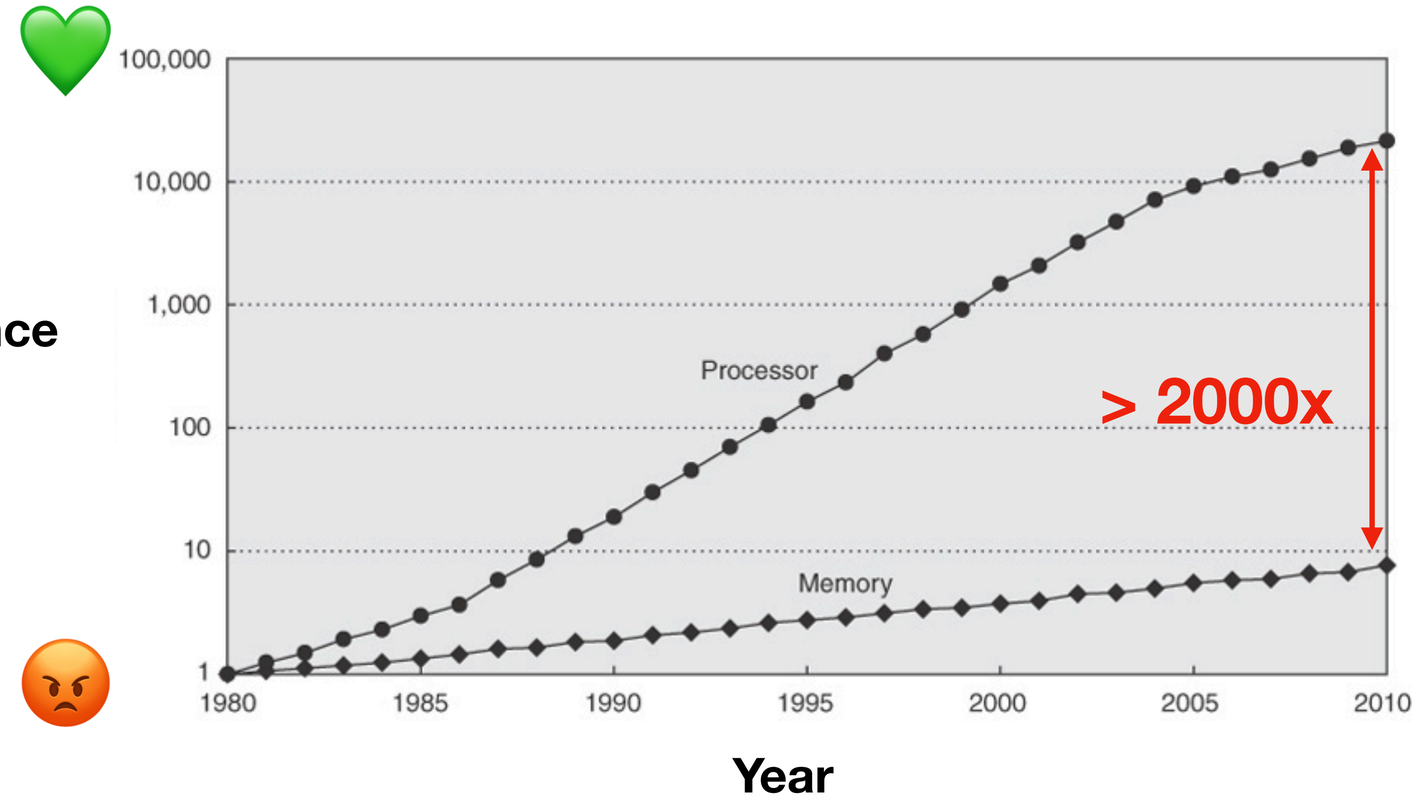
Use resources more efficiently, rather than using more resources

From Rich Vuduc

Motivating trends

How?		Driven by?
 **Parallelism**,	→	**Power**,
 **data locality**,	→	**memory**,
and **specialization**.	→	and **physics**.

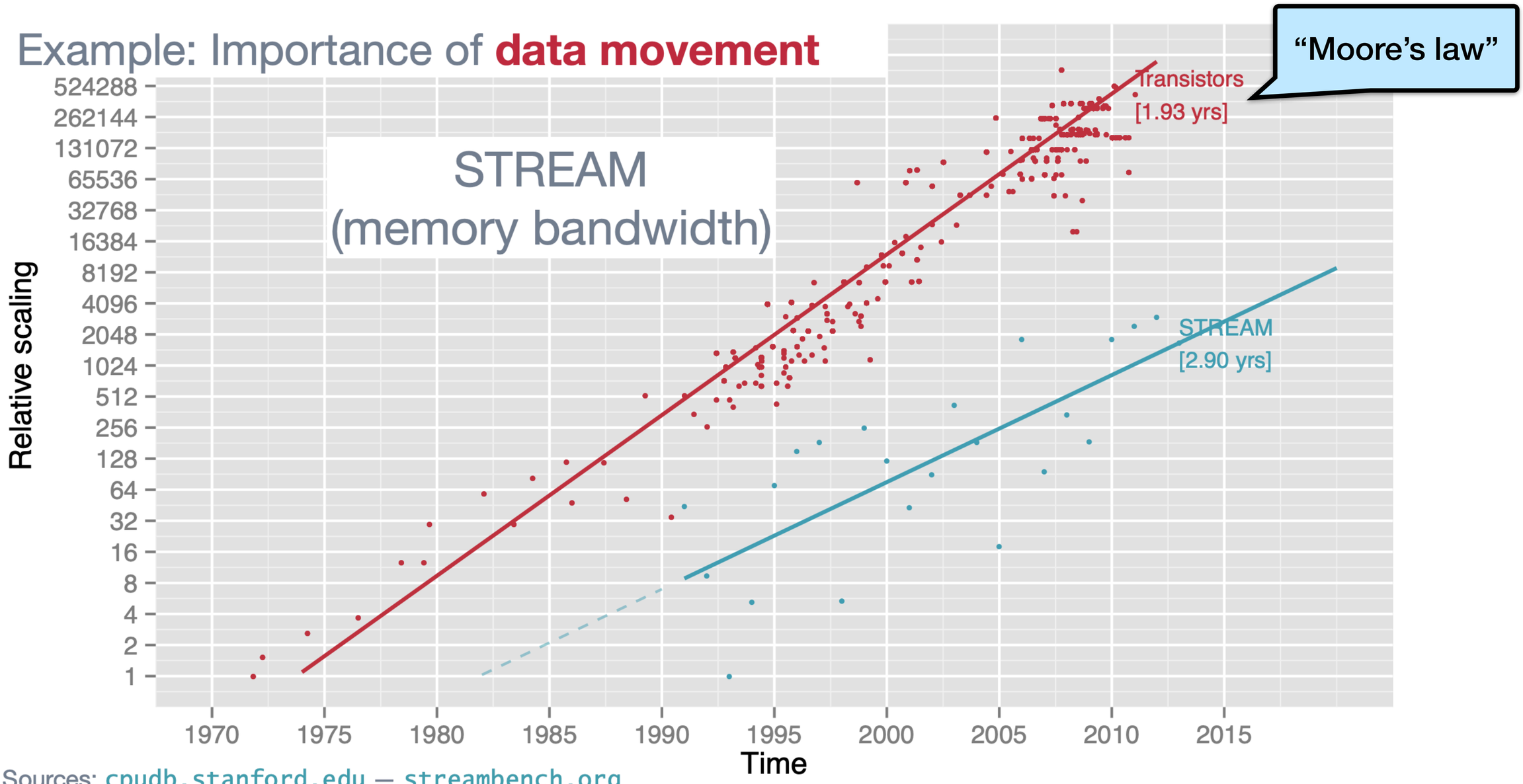
Data locality & the memory wall



Parallelization alone is not enough due to the **processor-memory gap**, or the exponentially increasing gap between CPU and memory speeds [Hennessy & Patterson 2011].

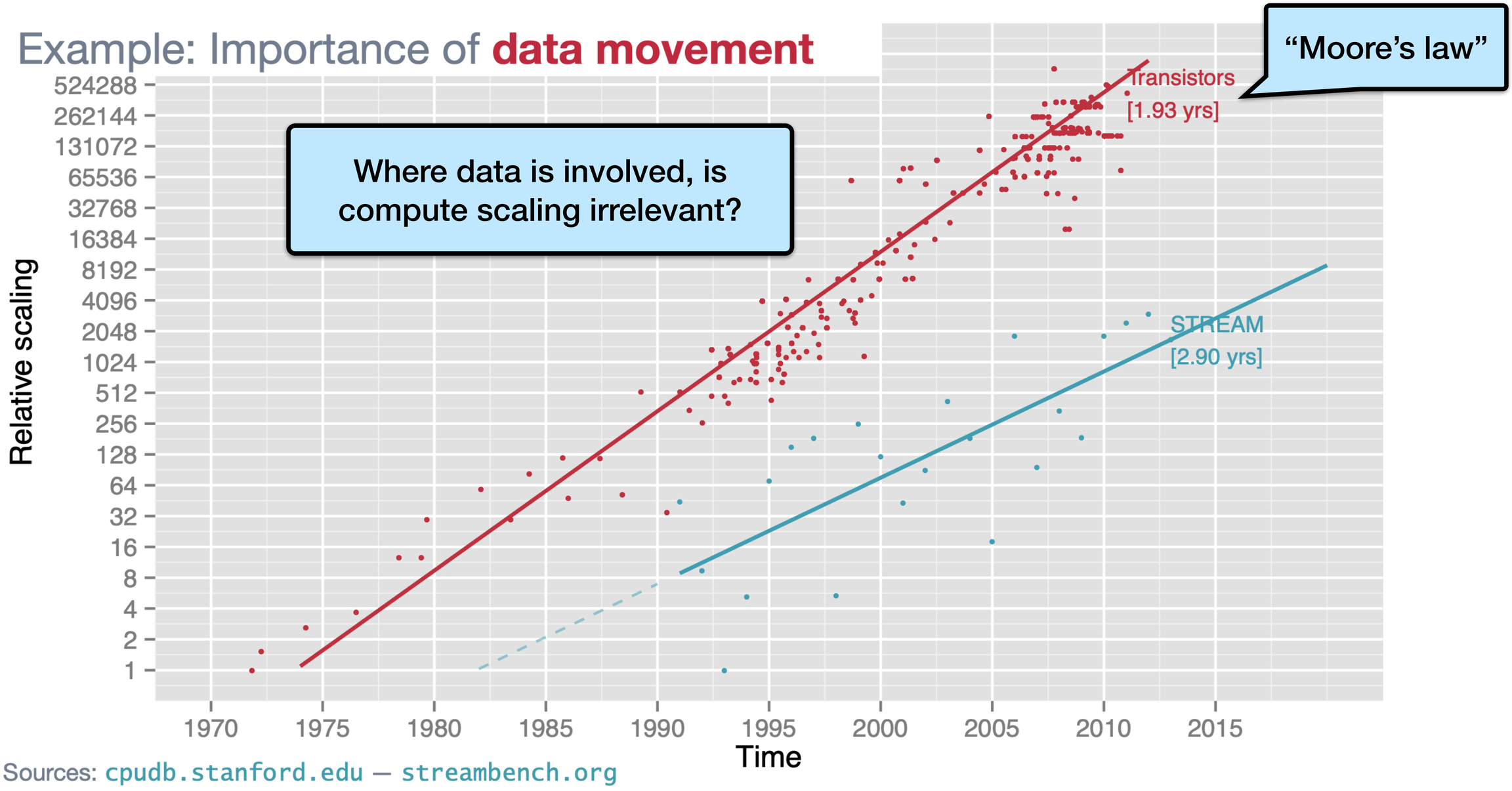
The memory wall, or the **gap between memory latency and latency of floating-point operations**, is increasing exponentially over time.

Example: Importance of **data movement**



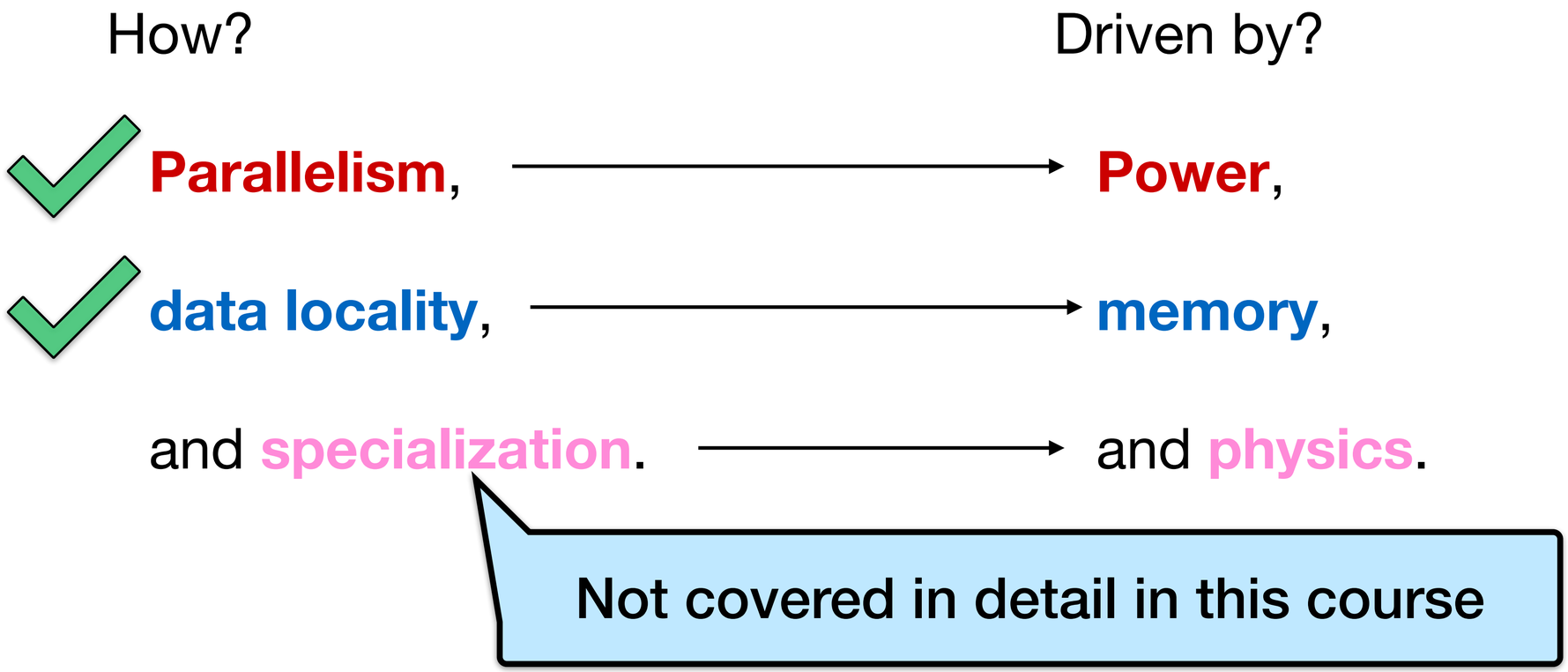
Sources: cpudb.stanford.edu — streambench.org

Example: Importance of **data movement**



Sources: cpudb.stanford.edu — streambench.org

Motivating trends



From Rich Vuduc

Example: **Specialization** (in hardware)

Understanding Sources of Inefficiency in General-Purpose Chips

Rehan Hameed[1], Wajahat Qadeer[1], Megan Wachs[1], Omid Azizi[1], Alex Solomatnikov[2], Benjamin C. Lee[1], Stephen Richardson[1], Christos Kozyrakis[1] and Mark Horowitz[1]

[1]Dept. of Electrical Engineering
Stanford University, Stanford, CA
{rhameed, wqadeer, wachs, oazizi,
bcclee, steveri, kozyraki, horowitz}@stanford.edu

[2]Hicamp Systems,
Menlo Park, CA
solomatnikov@gmail.com

~ 10x in energy (left) **& time** (right)
from "leaner" processor design

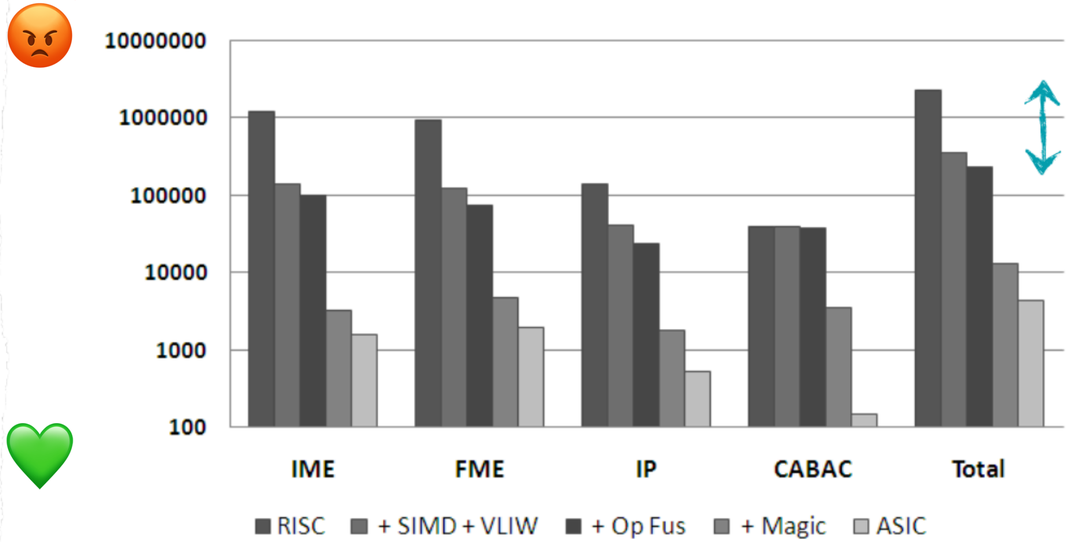


Figure 2. Each set of bar graphs represents energy consumption (µJ) at each stage of optimization for IME, FME, IP and CABAC respectively. Each optimization builds on the ones in the previous stage with the first bar in each set representing RISC energy dissipation followed by generic optimizations such as SIMD and VLIW, operation fusion and ending with "magic" instructions

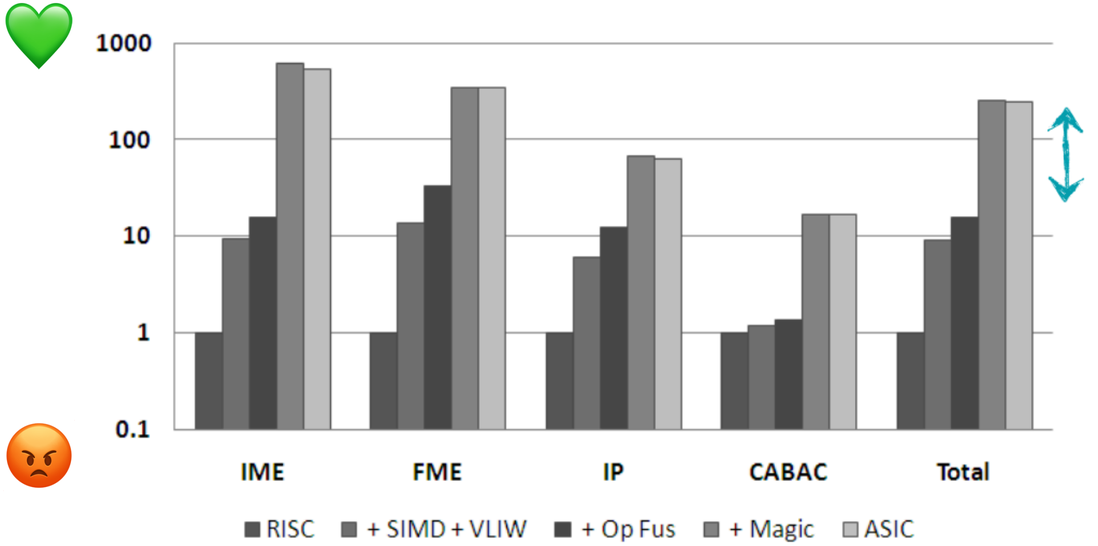


Figure 3. Each set of bar graphs represents speedup at each stage of optimization. Each optimization builds on those of the previous stage with the first bar in each set representing RISC speedup, followed by generic optimizations such as SIMD and VLIW, then operation fusion and finally "magic" instructions

occurring complex instruction subgraphs. Operation fusion is

From Rich Vuduc

Example: **Specialization** (in hardware)

Understanding Sources of Inefficiency in General-Purpose Chips

Rehan Hameed[1], Wajahat Qadeer[1], Megan Wachs[1], Omid Azizi[1], Alex Solomatnikov[2], Benjamin C. Lee[1], Stephen Richardson[1], Christos Kozyrakis[1] and Mark Horowitz[1]

[1]Dept. of Electrical Engineering
Stanford University, Stanford, CA
{rhameed, wqadeer, wachs, oazizi,
bcclee, steveri, kozyraki, horowitz}@stanford.edu

[2]Hicamp Systems,
Menlo Park, CA
solomatnikov@gmail.com

~ 10x in energy (left) & time (right)
from "leaner" processor design

~ 10x+ in energy (left) & time (right)
from aggressive customization

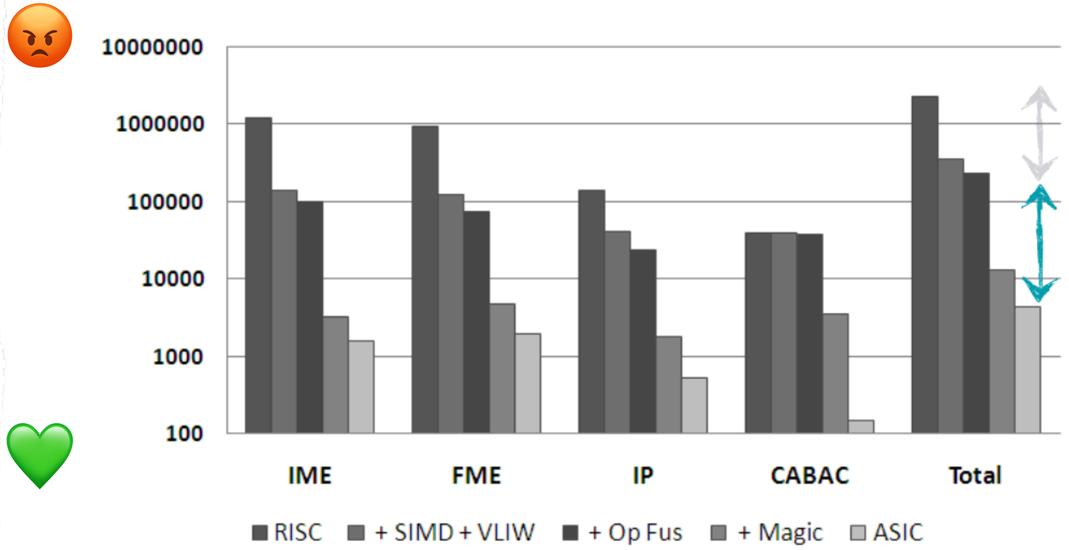


Figure 2. Each set of bar graphs represents energy consumption (µJ) at each stage of optimization for IME, FME, IP and CABAC respectively. Each optimization builds on the ones in the previous stage with the first bar in each set representing RISC energy dissipation followed by generic optimizations such as SIMD and VLIW, operation fusion and ending with "magic" instructions

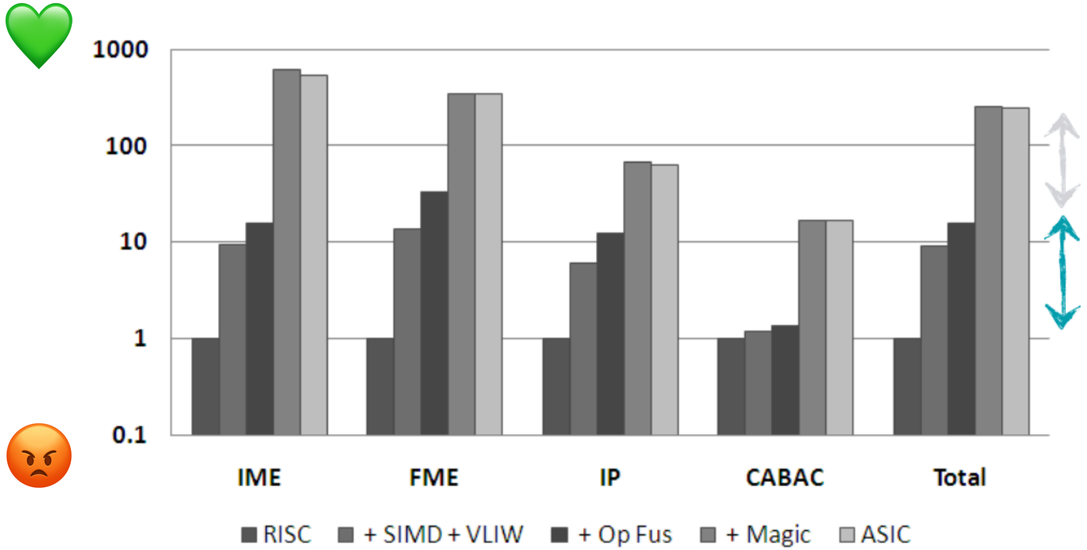
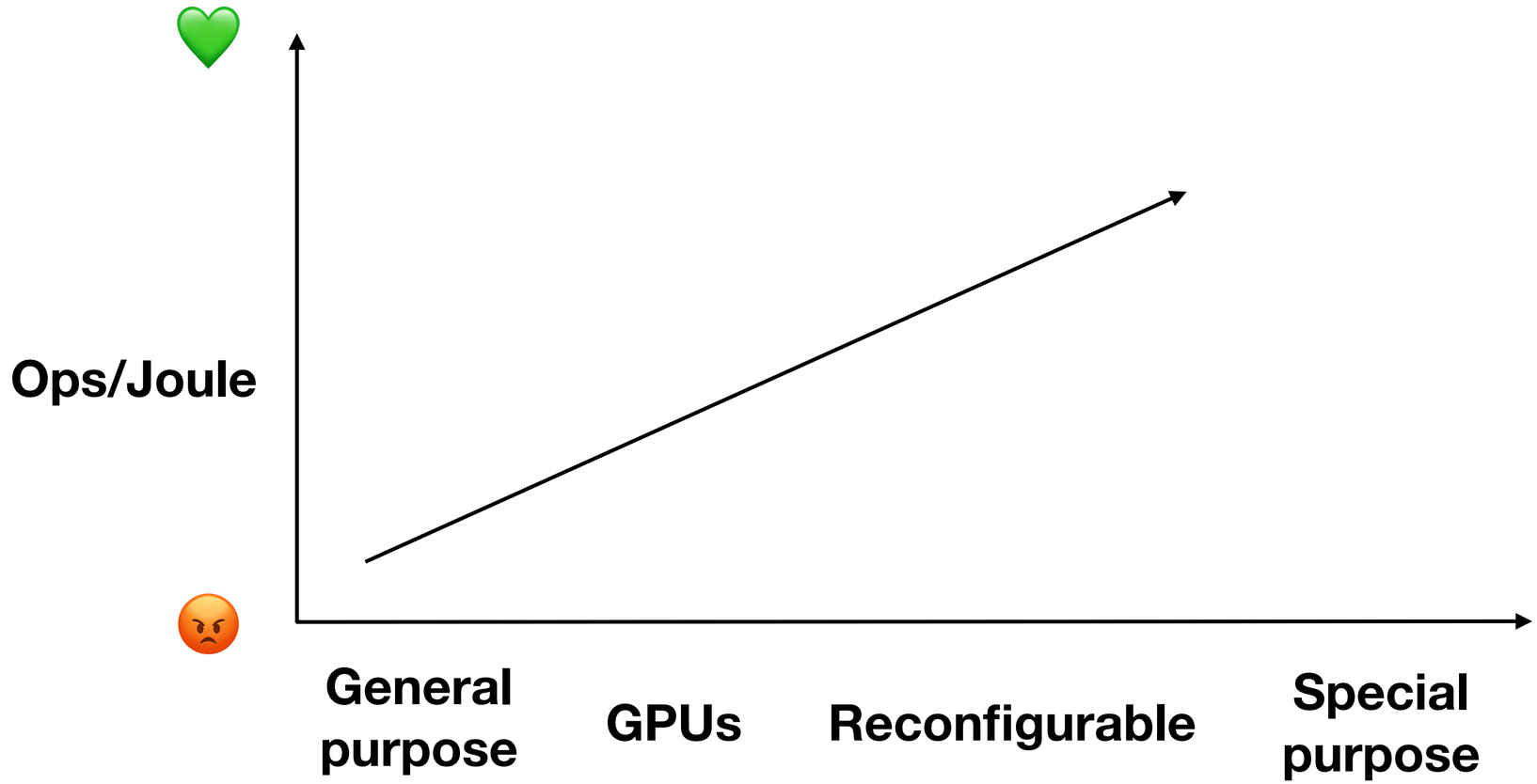


Figure 3. Each set of bar graphs represents speedup at each stage of optimization. Each optimization builds on those of the previous stage with the first bar in each set representing RISC speedup, followed by generic optimizations such as SIMD and VLIW, then operation fusion and finally "magic" instructions

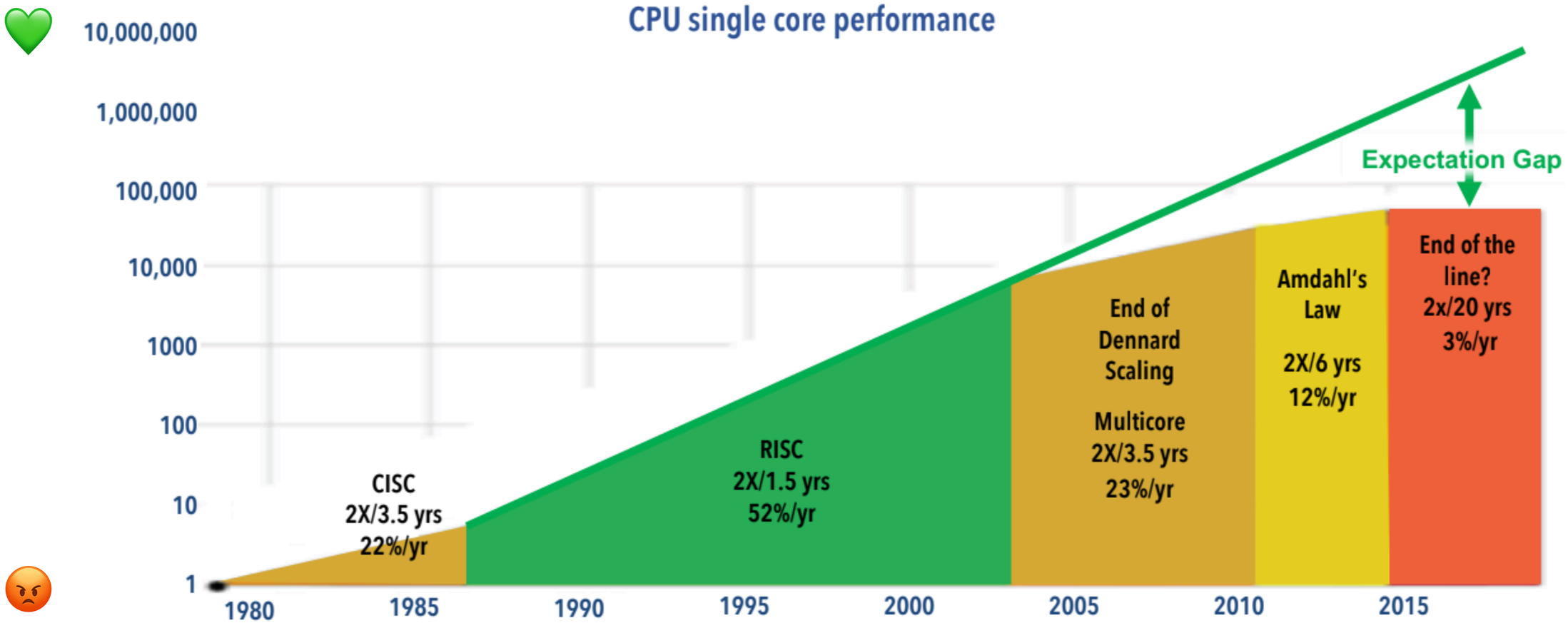
occurring complex instruction subgraphs. Operation fusion is

From Rich Vuduc

Specialization: End game for Moore's Law

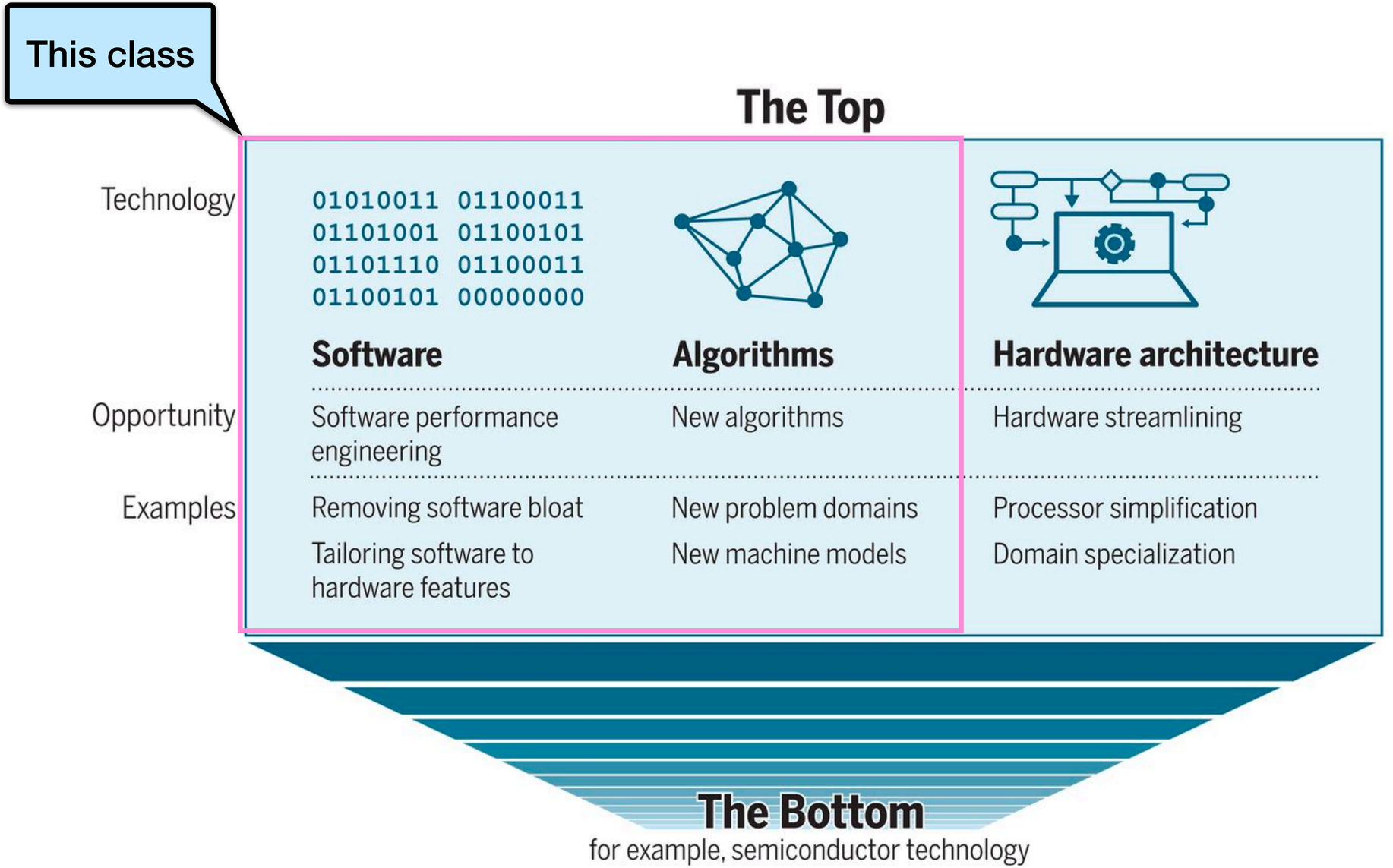


Traditional scaling is coming to an end



Hennessey/Patterson Turning Award Presentation: Golden Age of Computer Architecture Research

Performance gains after Moore's law ends



From "There's plenty of room at the Top: What will drive computer performance after Moore's law?" by Leiserson et al. (Science, 2020)