Scaling Laws in HPC and AI: Yesterday, Today and Tomorrow

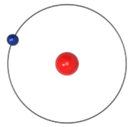
Institute of Science Tokyo
Rio Yokota

Salishan Conference on High Speed Computing
Salishan Coastal Lodge
April. 24, 2025

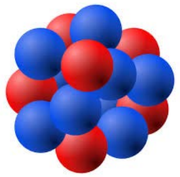
Scales of Nature

Fullerene molecule 10^{-9} [m]

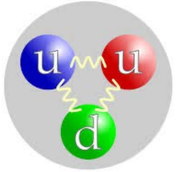
Hydrogen atom 10^{-10} [m]



Atomic nucleus 10^{-14} [m]



Proton 10^{-15} [m]



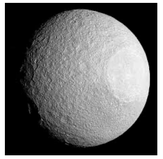
Bacteria 10^{-6} [m]

Tallest building 10^{3} [m]

Guitar 10^{0} [m]

Black ant 10^{-3} [m]

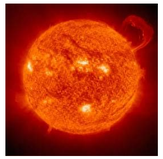
Tethys 10^{6} [m]



Earth 10^{7} [m]

Largest star 10^{12} [m]

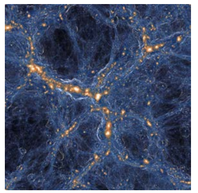
Sun 10^{9} [m]



Largest black hole 10^{15} [m]

Supernova remnant 10^{18} [m]

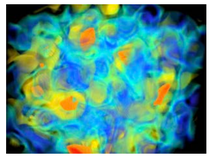
Galaxy filament 10^{24} [m]

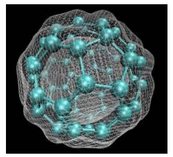


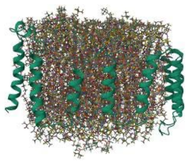
Milky way galaxy 10^{21} [m]

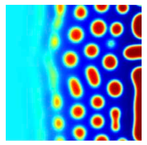
Observable universe 10^{27} [m]

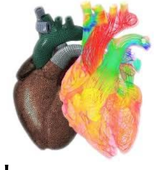
10^{-15} 10^{-12} 10^{-9} 10^{-6} 10^{-3} 10^{0} 10^{3} 10^{6} 10^{9} 10^{12} 10^{15} 10^{18} 10^{21} 10^{24} 10^{27} [m]

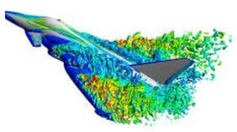


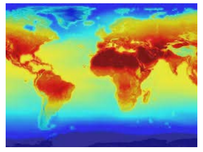


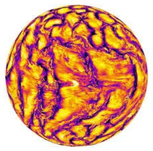


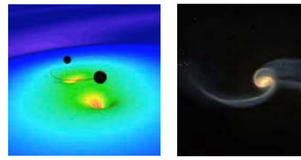


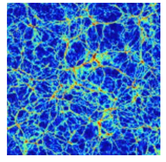












LQCD DFT MD Phase Field FEM/FVM/FDM N-body/SPH

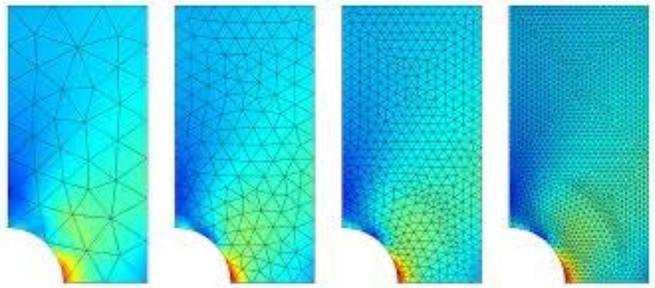
Mesh Particle Mesh Particle

How to Model Complex (Non-linear) Systems?

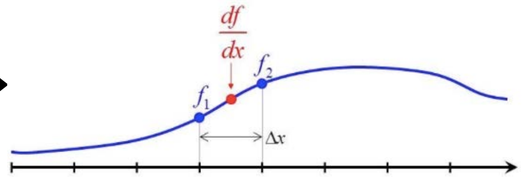
Mesh

Piece-wise linear functions

$$\frac{\partial f}{\partial x} = \frac{f_2 - f_1}{\Delta x}$$



https://www.comsol.jp/multiphysics/mesh-refinement



$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

Millennium Prize Problems

- Birch and Swinnerton-Dyer conjecture
- Hodge conjecture
- Navier–Stokes existence and smoothness
- P versus NP problem
- Poincaré conjecture (solved)
- Riemann hypothesis
- Yang–Mills existence and mass gap

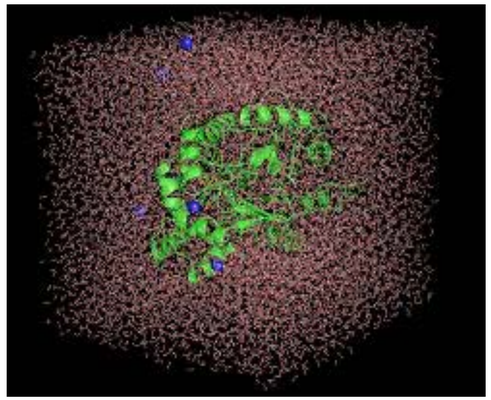
$$Ax = b$$

Basic linear algebra
Clear cut API

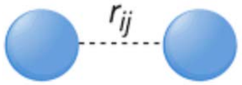
Particle

Pair-wise linear functions

$$F = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}^2}$$



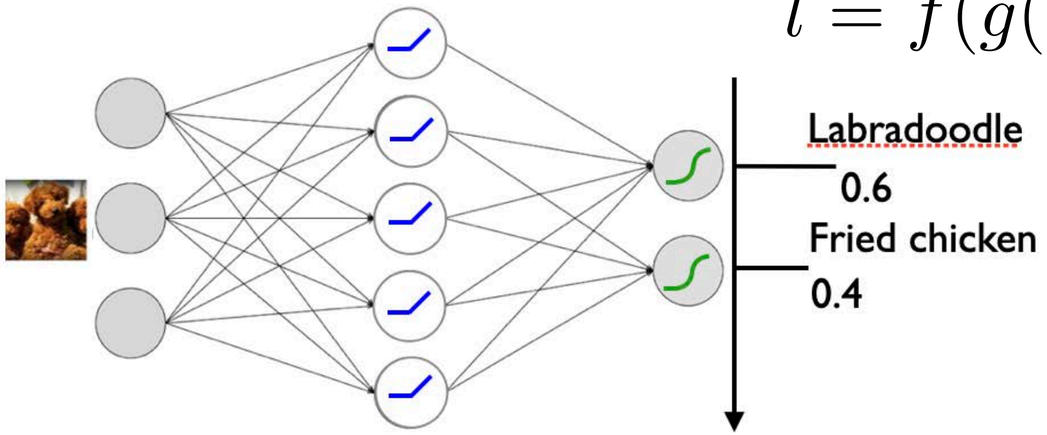
https://qstatix.co.in/molecular-dynamics-simulations/



Neural Network

Composite (mostly) linear functions

$$l = f(g(h(x)))$$



Forward propagation

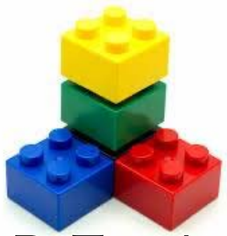
$$x_0 \quad x_1 = ReLU(z_0) \quad p = softmax(z_1)$$

$$z_0 = W_0 x_0 \quad z_1 = W_1 x_1 \quad l = -\log p$$

Backward propagation

$$\frac{\partial x_1}{\partial z_0} * \frac{\partial z_1}{\partial x_1} * \frac{\partial l}{\partial z_1} = \frac{\partial l}{\partial \theta}$$

$$\frac{\partial x_1}{\partial z_0} \otimes \frac{\partial z_0}{\partial W_0} \qquad \frac{\partial z_1}{\partial W_1}$$



PyTorch

The Bitter Lesson

Rich Sutton

March 13, 2019

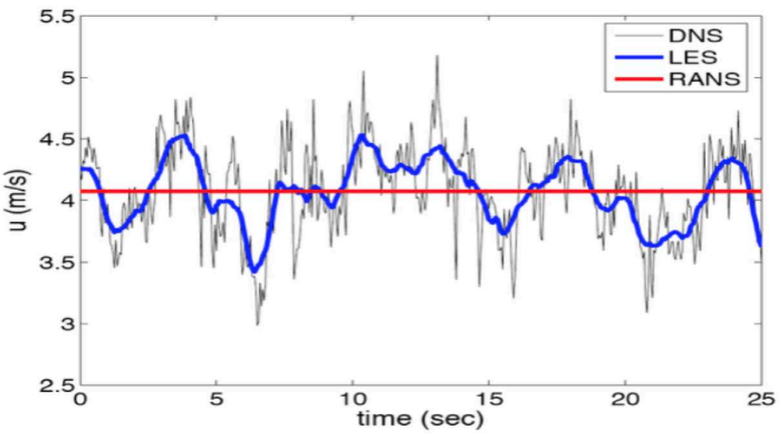
The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation.

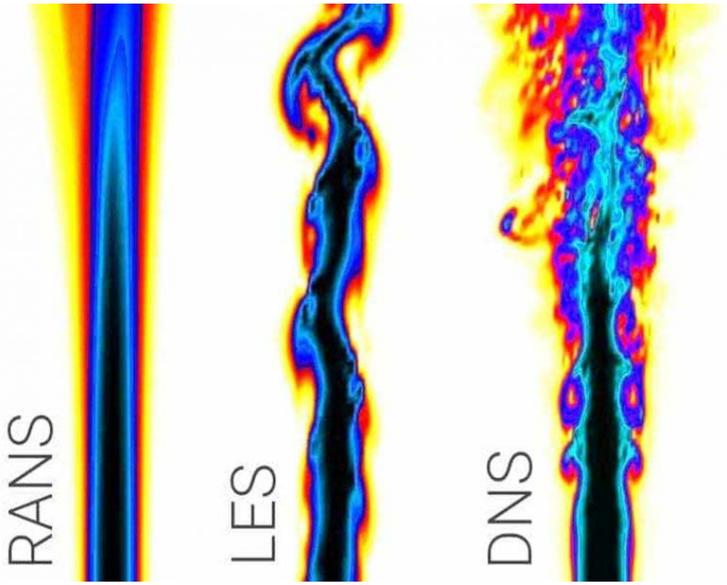
Turbulence Modeling

DNS: Direct Numerical Simulation
→Solves the Navier-Stokes equation directly (no modeling)

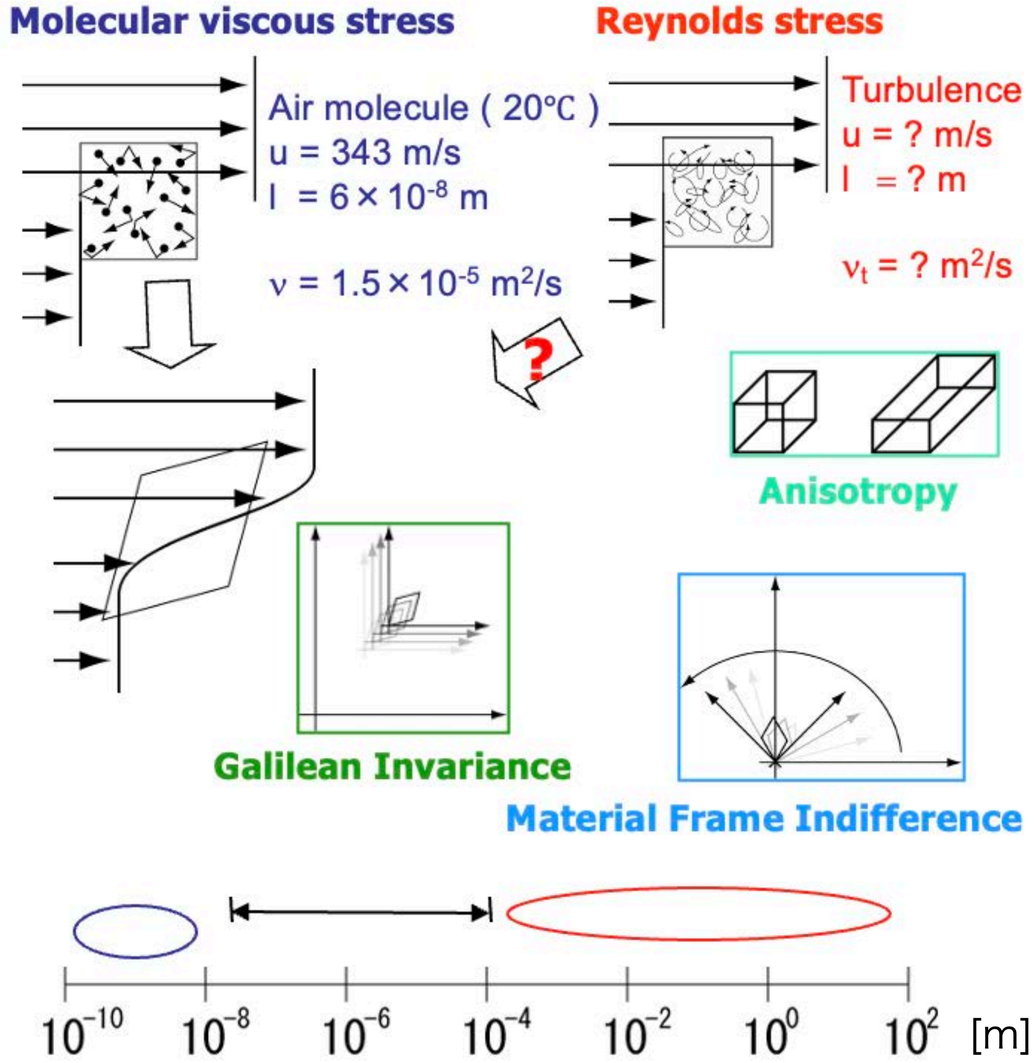
LES: Large Eddy Simulation
→Applies a spatial filter and models the tiny eddies

RANS: Reynolds Averaged Navier-Stokes
→Models the time-averaged flow





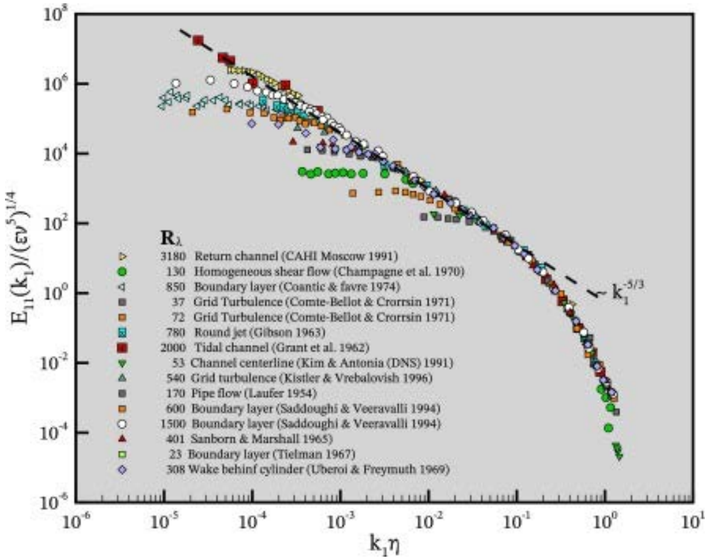
https://gibbs.science/les/lectures/lecture_05.pdf



https://www.idealsimulations.com/resources/turbulence-models-in-cfd/

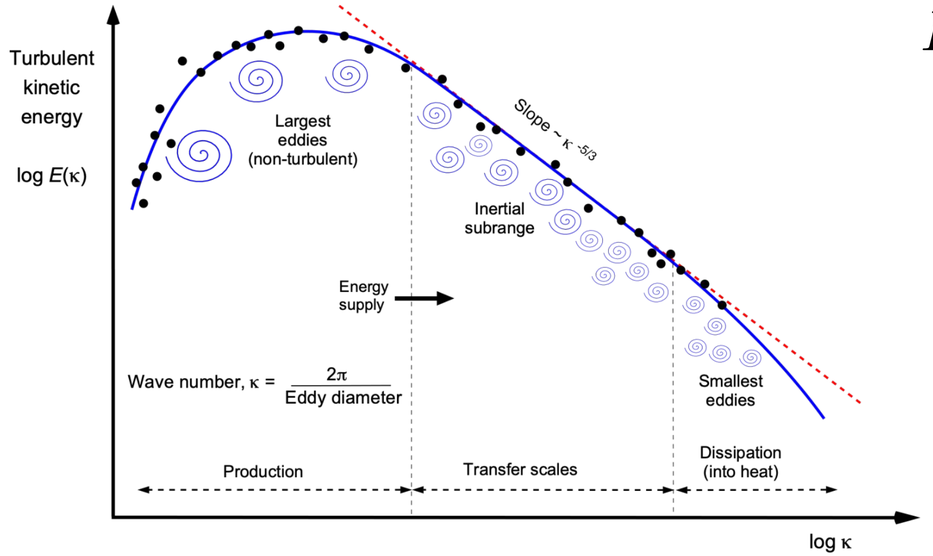
Scaling Laws in Turbulence

Experiment (1st paradigm)



https://www.sciencedirect.com/topics/engineering/turbulent-kinetic-energy-spectrum

Theory (2nd paradigm)

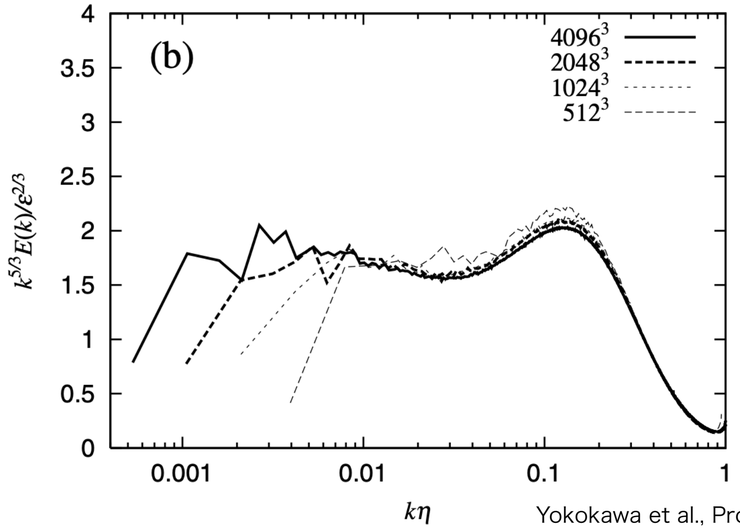


https://eaglepubs.erau.edu/introductiontoaerospaceflightvehicles/chapter/turbulent-flows/

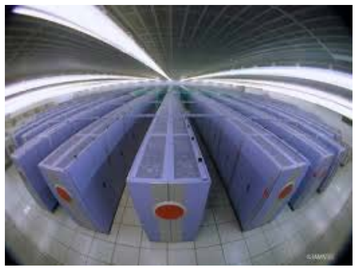
Power law

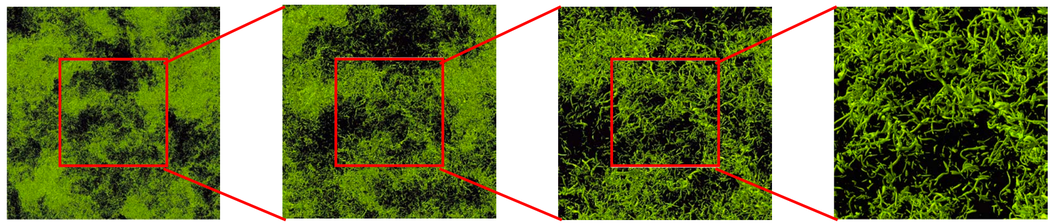
$$E(k) = C_K \epsilon^{2/3} k^{-5/3}$$

Simulation (3rd paradigm)



System: Earth Simulator
Method: Spectral DNS (3-D FFT)
Resolution: 4096^3
Performance: 16.4-Tflops



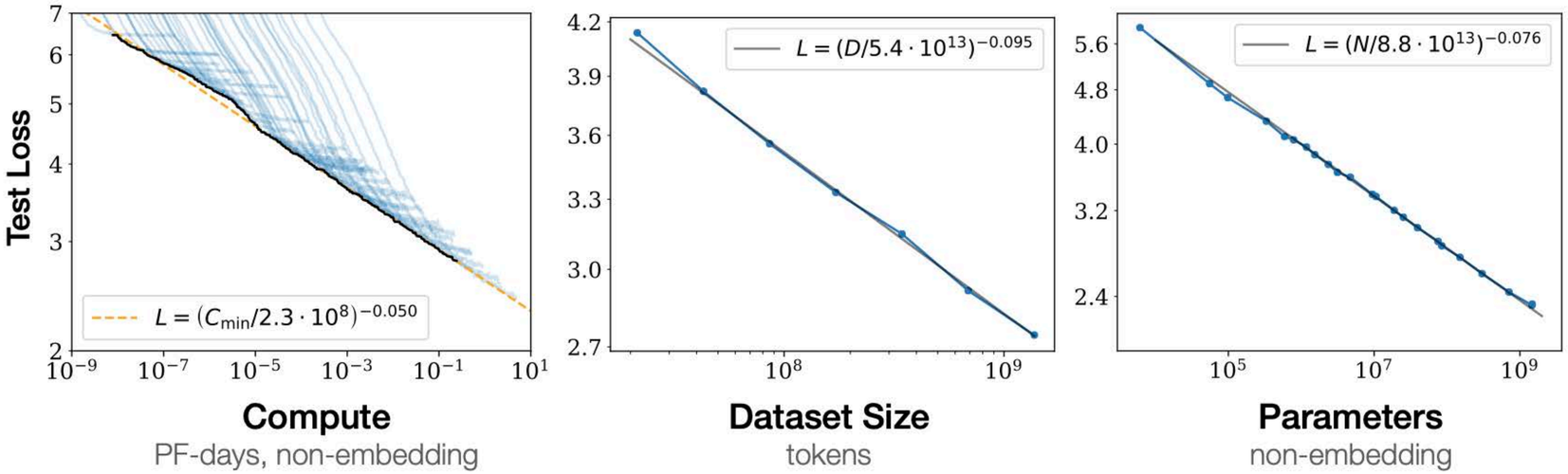


Yokokawa et al., Proc. of SC02 (Gordon Bell prize), https://ieeexplore.ieee.org/document/1592886

Scaling Laws in Deep Learning

Kaplan's scaling law

https://arxiv.org/abs/2001.08361



Power law

N: model size [parameters]

D: data size [tokens]

Kaplan2020: $D \propto N^{0.73}$

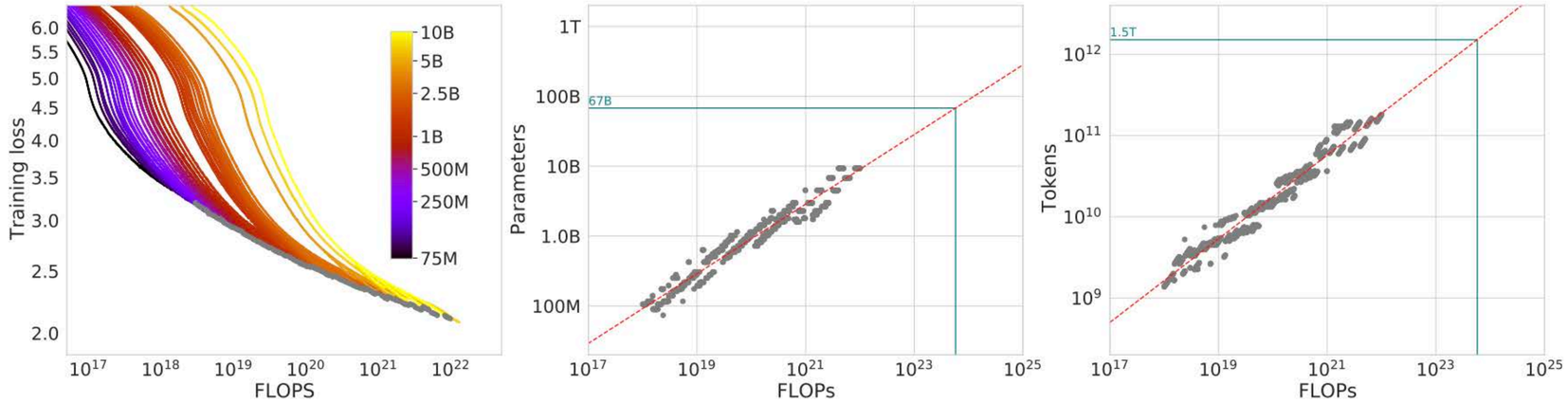
→Fixed LR schedule

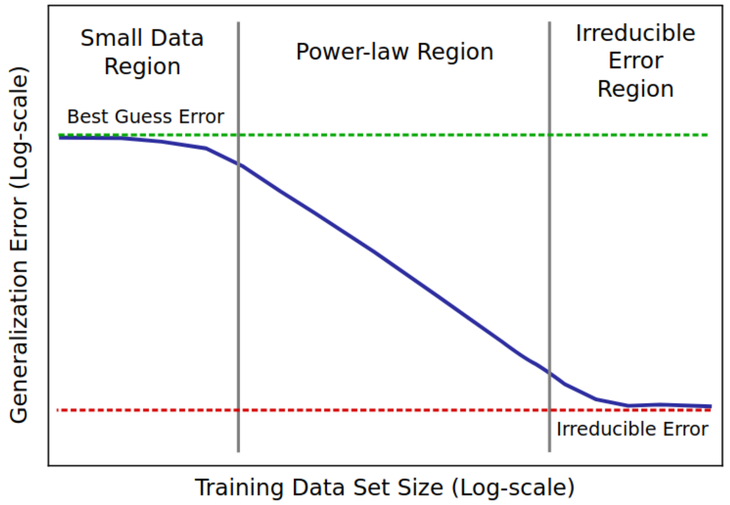
Hoffman2022: $D \propto N$

→Optimal LR schedule

Hoffman's scaling law (Chinchilla law)

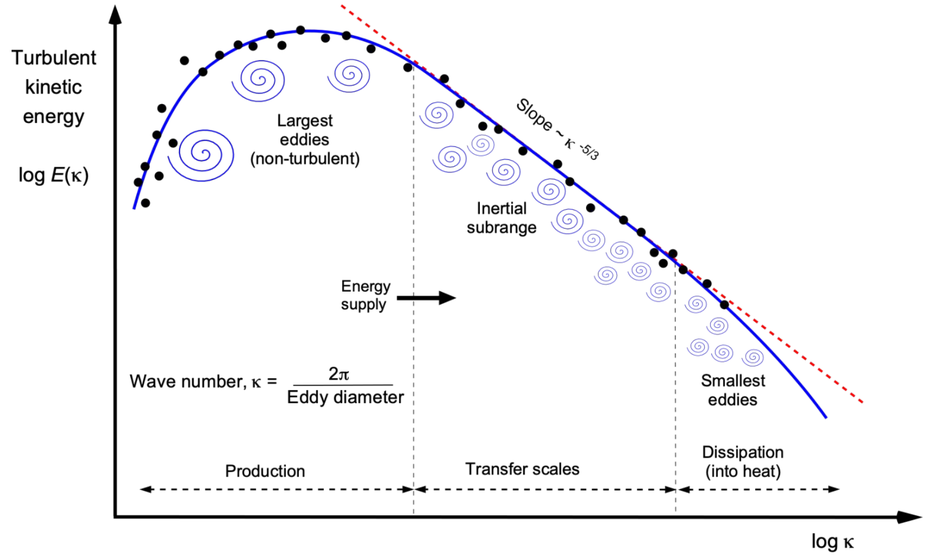
https://arxiv.org/abs/2203.15556

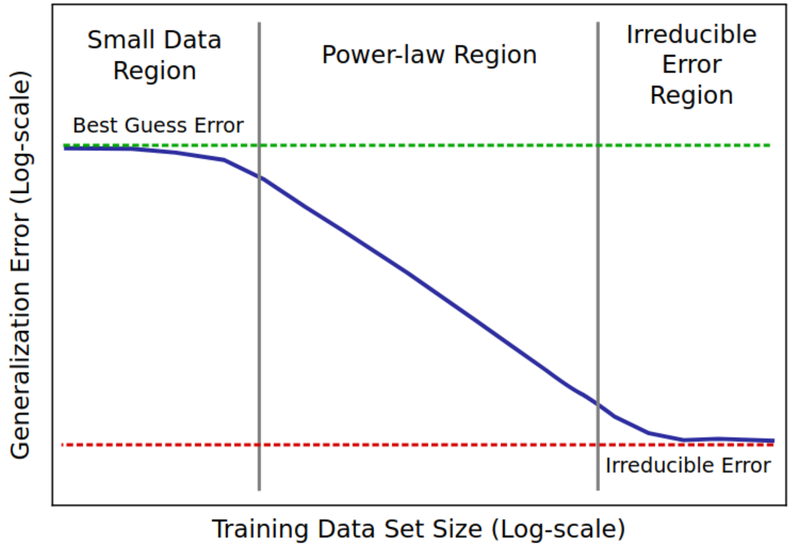




https://epochai.org/blog/scaling-laws-literature-review

Similarities and Differences of Scaling Laws





Similarities

Justified the largest supercomputer in 2002 ⟷ Justifying the largest GPU clusters in 2025

Actual CFD applications have many other issues ⟷ Actual AI applications have many other issues

・Near wall ・Multi-phase ・Multi-physics ・Data quality/quantity ・Copyright ・Alignment

Differences

Investigation was driven by scientific interest ⟷ Investigation was driven by commercial interest

Validation between the 1st, 2nd, 3rd paradigm ⟷ Asymptotic behavior of the 4th paradigm

In 2002 Moore's law was still alive ⟷ In 2025 Moore's law is dying/dead

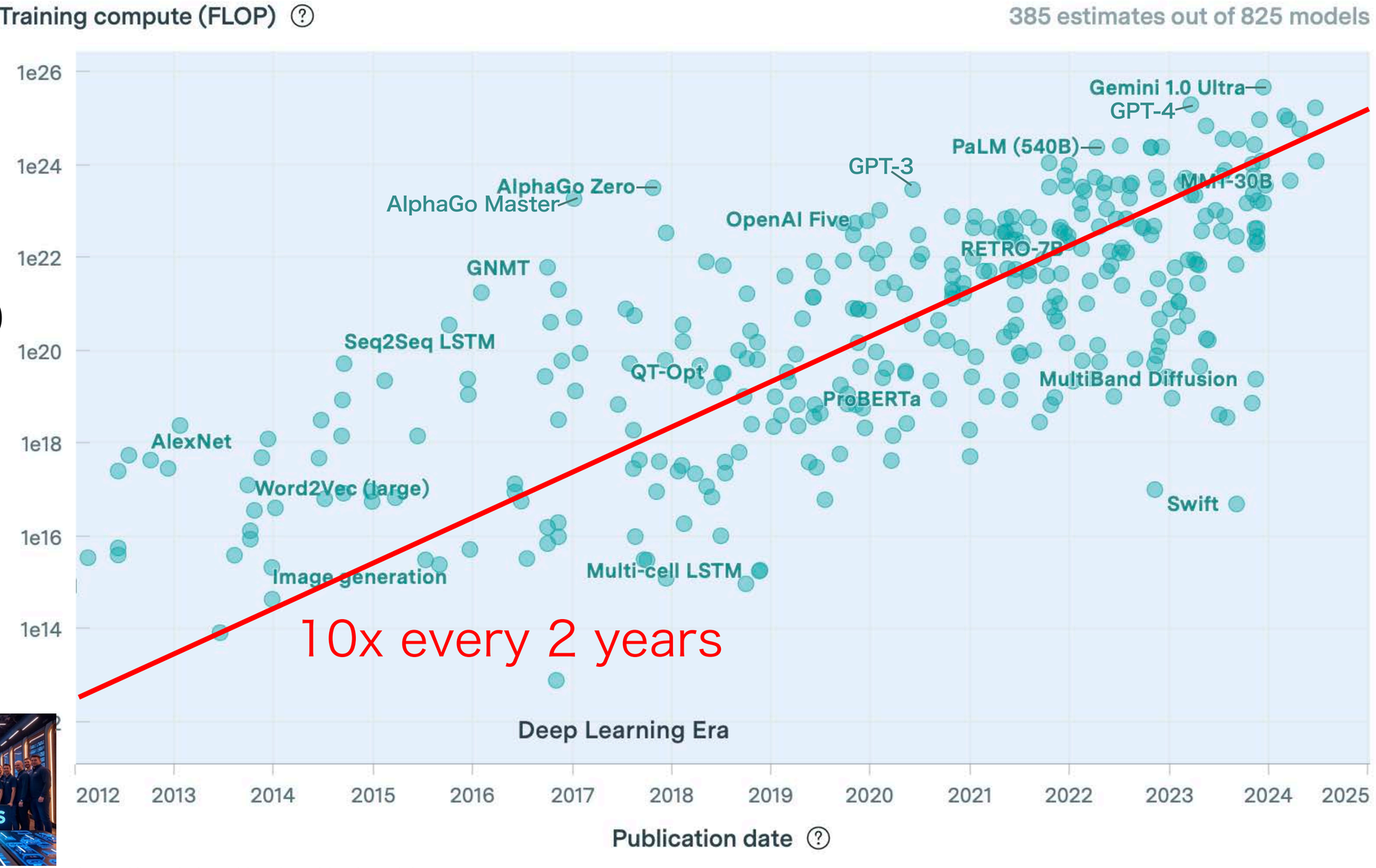
The ~~Scaling~~ Spending Law

2012: **AlexNet**
$500 GPU x 2
= **$1K**
5 days of training

2022: **GPT-4** (speculation)
$8K GPU x 25,000
= **$200M**
90 days of training

Any growth beyond
Moore's law is achieved
only by spending more
2024: $10B
2026: $50B
2028: $250B
2030: $1.25T

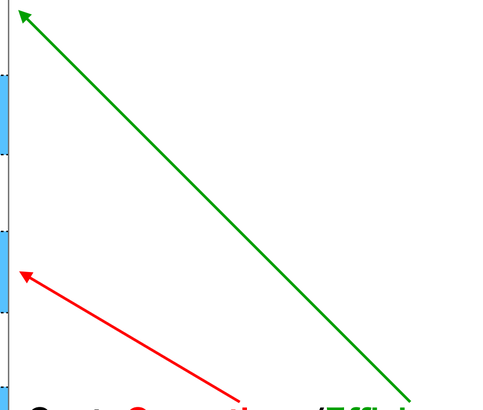
100,000 x H100



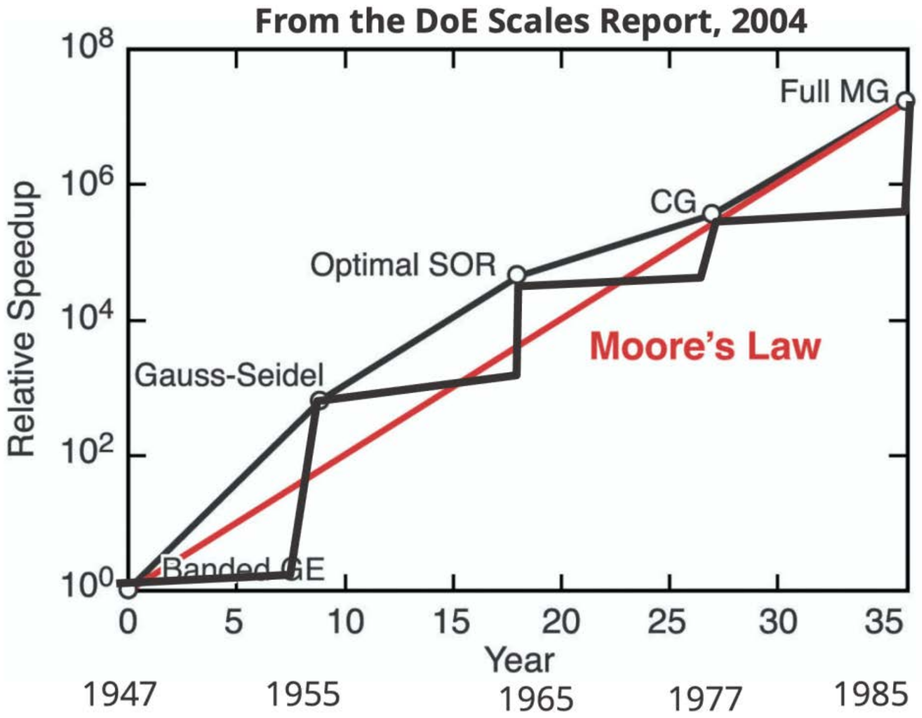
https://epochai.org/data/notable-ai-models

OpenAI vs. Meta vs. DeepSeek

		OpenAI	Meta	DeepSeek
Model	Name	GPT-4	Llama3.1 405B	DeepSeek-V3
	Total Parameters	1.8T	405B	671B
	Active Parameters	280B	405B	37B
Data	Tokens	13T	15T	14.8T
	Open?	No	Yes	Yes
GPUs	GPU Type	A100	H100	H800
	#GPUs	25,000	16,384	2,048
	FLOP/s/GPU	312 TFLOP/s	989 TFLOP/s	989 TFLOP/s
	% of peak	32.5%	48%	17.8% (8bit) 35.6% (16bit)
Cost	Days to train	90 days	54 days	55 days
	GPU Hours	60M	21M	2.6M
	FLOPs	2.2E+25	3.6E+25	3.3E+24
	Price on AWS	$54M	$89M	$11.2M
	$2/GPU Cloud	$27M	$44.5M	$5.6M



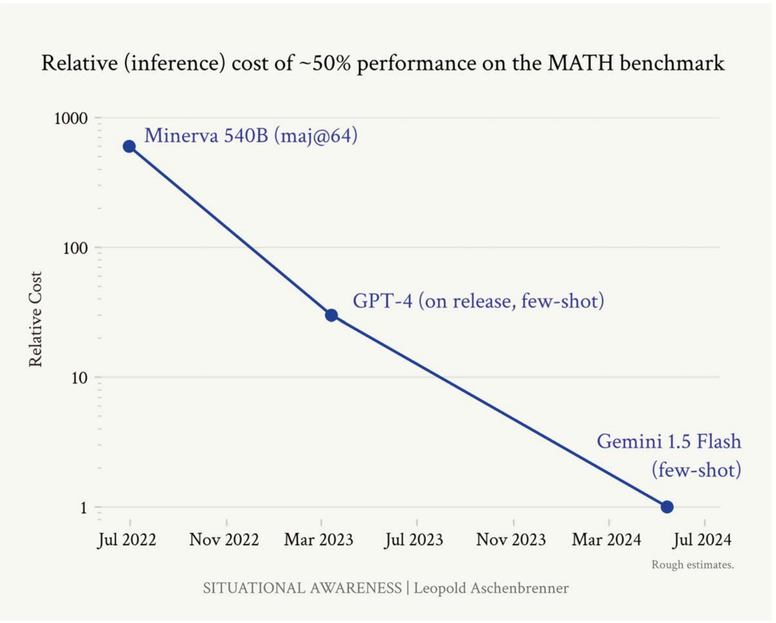
It’s Not All Brute Force Computing



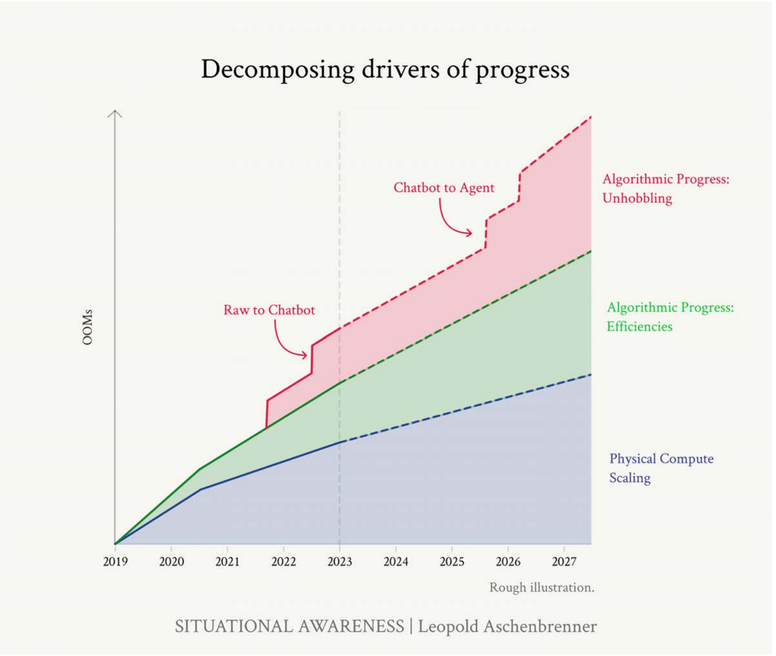
Algorithmic improvement in linear solvers has brought as much speedup as Moore’s law

“With great computational power comes great algorithmic responsibility” — Longfei Gao

Inference cost to achieve the same performance on MATH benchmark decreased 600x in 2 years



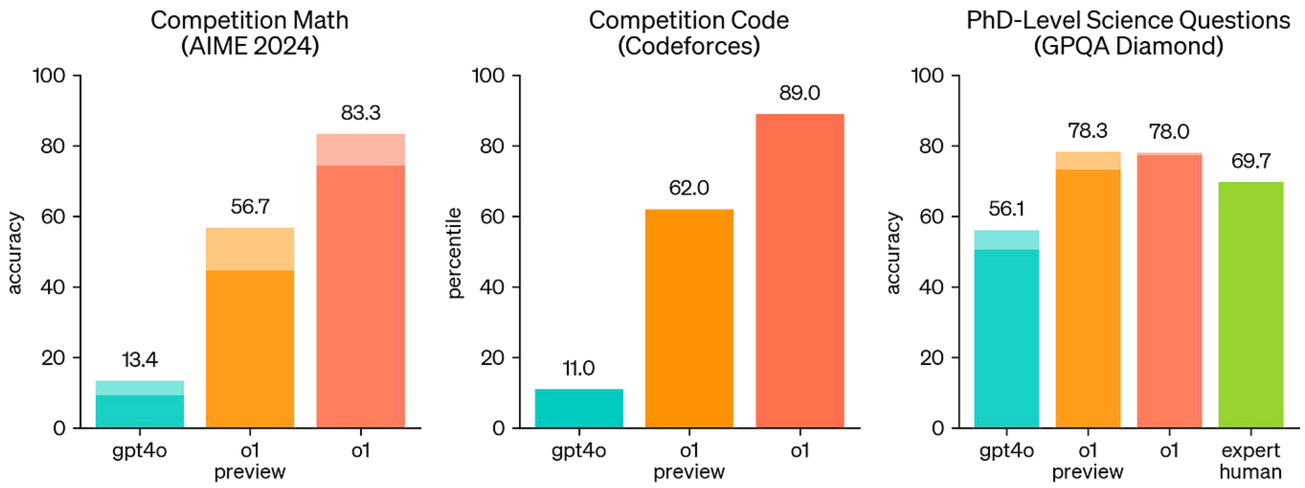
There is much more to be gained through algorithmic progress via unhobbling

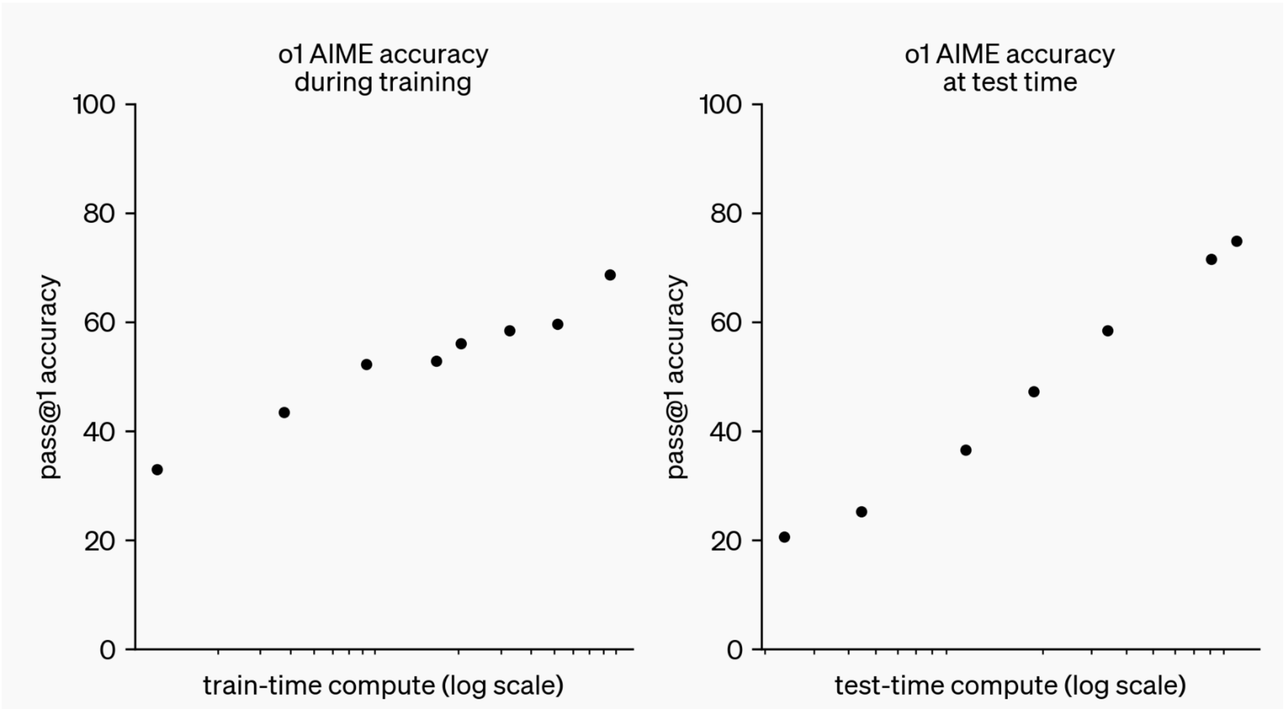


https://situational-awareness.ai/from-gpt-4-to-agi/

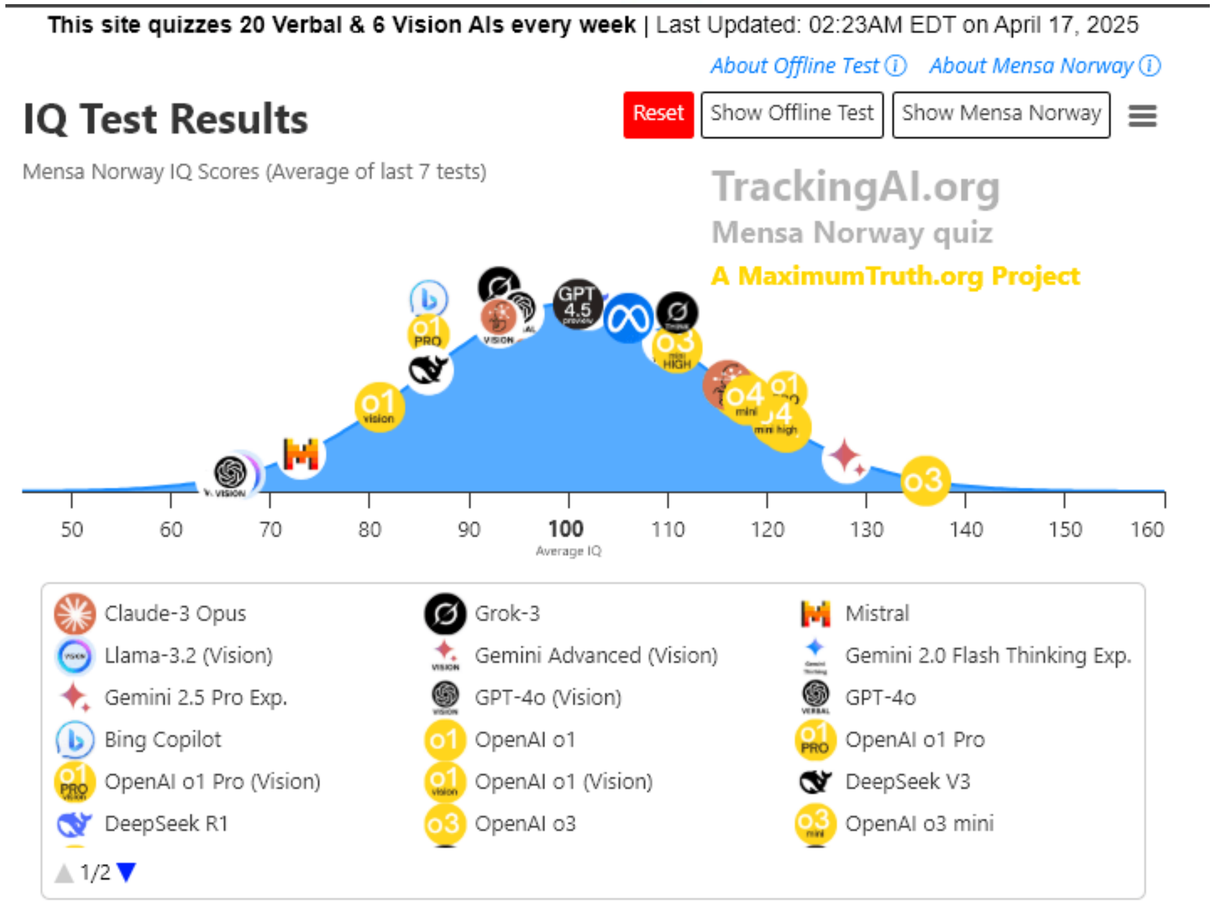
Unhobbling: The Test-Time Scaling Law

OpenAI o1





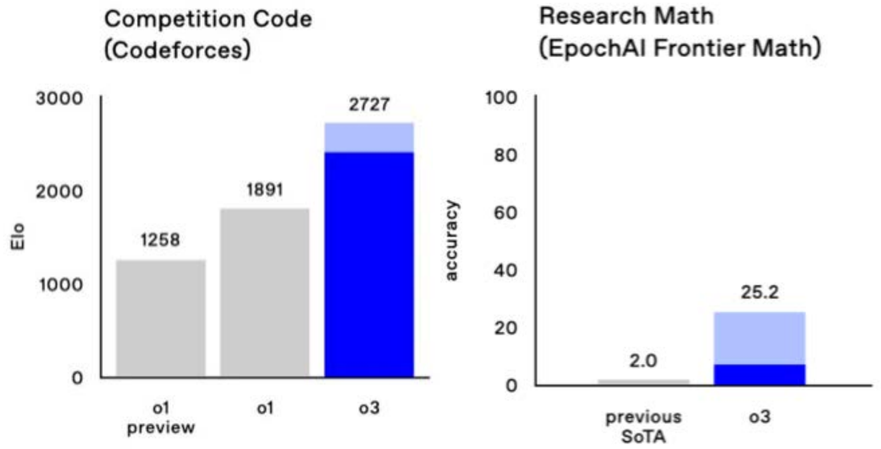
https://openai.com/index/learning-to-reason-with-llms/



I have played a little bit with OpenAI's new iteration of #GPT, GPT-o1, which performs an initial reasoning step before running the LLM. It is certainly a more capable tool than previous iterations, though still struggling with the most advanced research mathematical tasks.

https://mathstodon.xyz/@tao/113132502735585408

Implications of Test-time Scaling Laws



Implications

We need more difficult evals (benchmarks)
→Our models are only as good as our evals

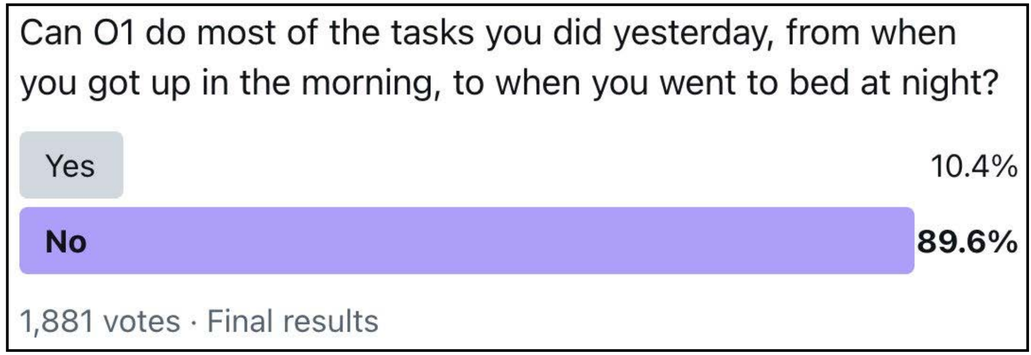
Test-time scaling is crucial for “AI for Science”
→With the excellent math and coding skills on the horizon, giant leaps in science will become possible
→On which data should we pretrain vs. learn in-context

Caveats

Test-time scaling will not make the model learn something it already doesn’t know
→You need to have a strong base model for the chain-of-thoughts to lead anywhere
→What is the minimum model/data size that leads to test-time scaling / reasoning?

Test-time scaling requires exponential cost for inference
→Training cost gets amortized, but inference cost doesn’t

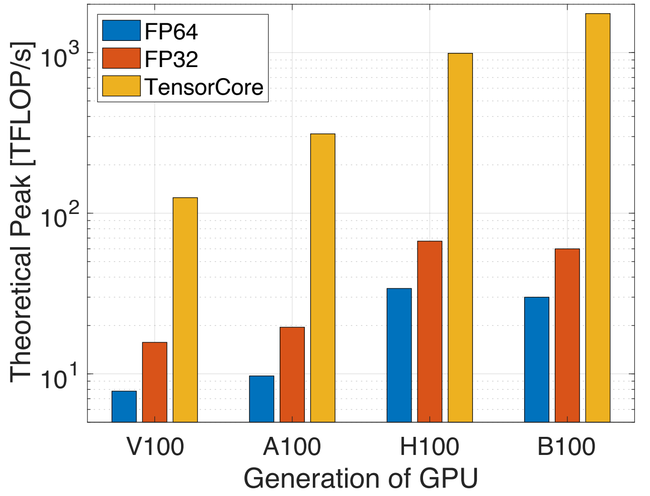
Test-time scaling alone cannot solve all real world tasks
→Orchestration of agents (both online and real world)



Implications for “AI for Science”

Hardware design is driven by the AI market

- Use TensorCores in your code
  - Recovering precision
  - Reformulate into GEMM

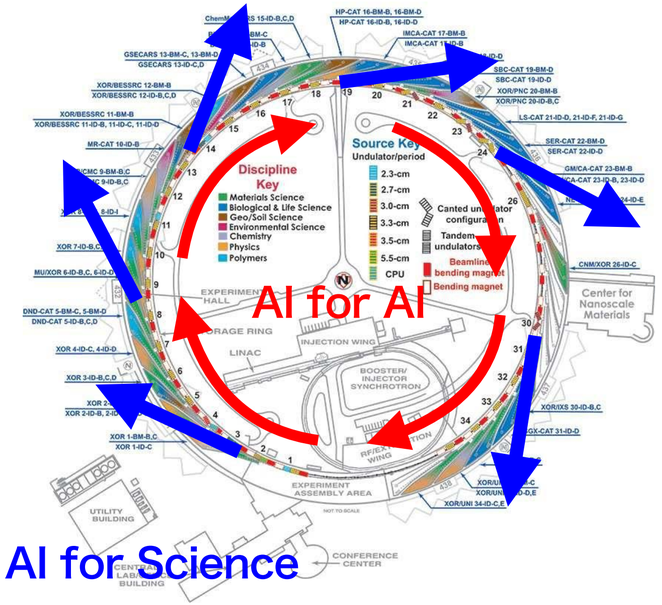


Myths of “AI for Science”

- AI companies are not interested in science
- PetaBytes of raw scientific data can be used for training
- Models cannot learn from synthetic data

Truths of “AI for Science”

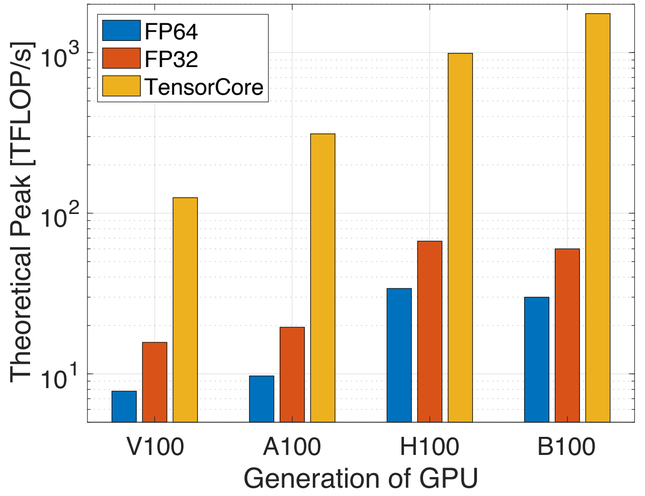
- Test-time scaling is crucial for “AI for Science”
- You need to have a strong base model for the chain-of-thoughts to lead anywhere
- Progress depends on how close we can get to the “AI for AI” cycle



AI Hardware for Science

Hardware design is driven by the AI market

- Use TensorCores in your code
  - Recovering precision
  - Reformulate into GEMM



Recovering FP32 from FP16 Tensor Cores

- Compensated summation + rounding with RN

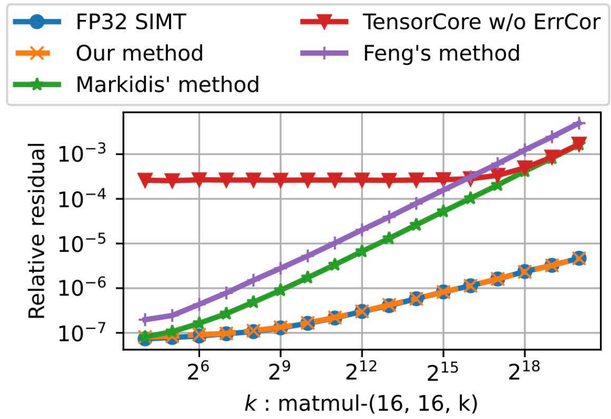
Recovering FP64 from INT8 Tensor Cores

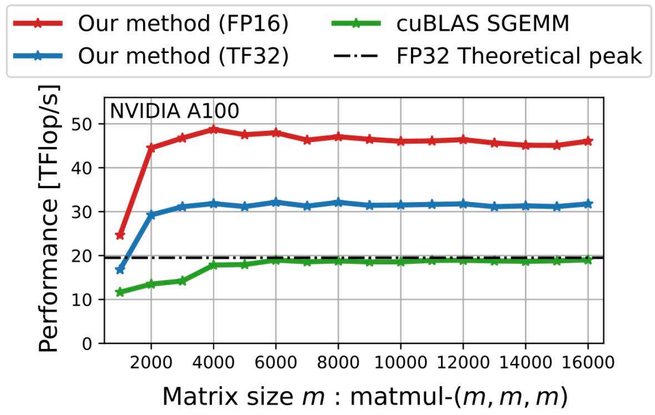
- Ozaki scheme

Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance

Hiroyuki Ootomo [1] and Rio Yokota [2]

https://arxiv.org/abs/2203.03341

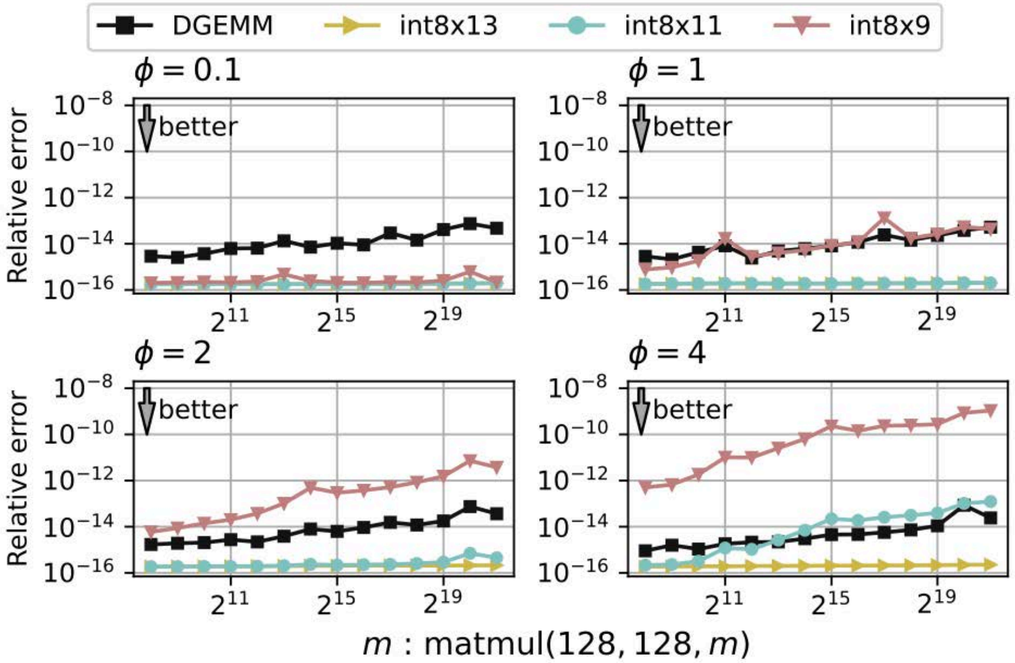




DGEMM on integer matrix multiplication unit

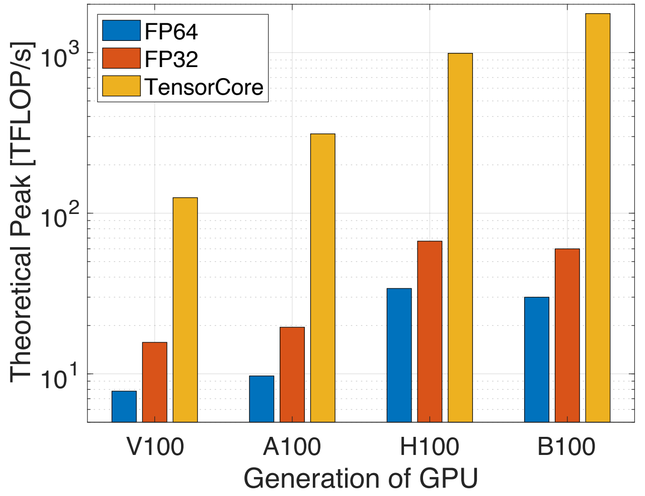
Hiroyuki Ootomo, Katsuhisa Ozaki, Rio Yokota

https://arxiv.org/abs/2306.11975



AI Hardware for Science

Hardware design is driven by the AI market →



- Use TensorCores in your code
  - Recovering precision
  - Reformulate into GEMM

Stencil

- Toward accelerated stencil computation by adapting tensor core unit on GPU [ICS'22]
- ConvStencil: Transform Stencil Computation to Matrix Multiplication on Tensor Cores [PPoPP'24]
- LoRAStencil: Low-Rank Adaptation of Stencil Computation on Tensor Cores [SC'24]

SpMM

- Efficient tensor core-based GPU kernels for structured sparsity under reduced precision [SC'21]
- Probing the Efficacy of Hardware-Aware Weight Pruning to Optimize the SpMM Routine on Ampere GPUs [PACT'22]
- Efficient Quantized Sparse Matrix Operations on Tensor Cores [SC'22]
- High Performance Unstructured SpMM Computation Using Tensor Cores [SC'24]

FFT

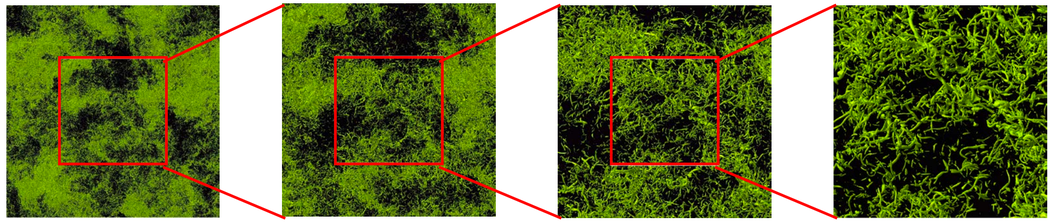
- Accelerating Fourier and Number Theoretic Transforms using Tensor Cores and Warp Shuffles [PACT'21]
- Accelerating non-power-of-2 size Fourier transforms with GPU Tensor Cores [IPDPS'21]
- tcFFT: A Fast Half-Precision FFT Library for NVIDIA Tensor Cores [Cluster'21]

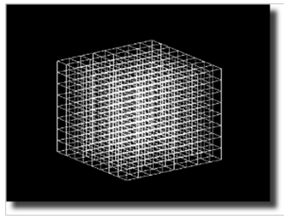
Multigrid

- AmgT: Algebraic Multigrid Solver on Tensor Cores [SC'24]

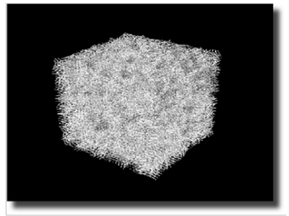
list shared by Mohamed Wahib (RIKEN)

Adapting Applications to Specialized Hardware

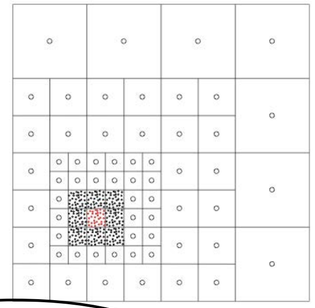




Spectral DNS
FFT



Vortex Method
N-body

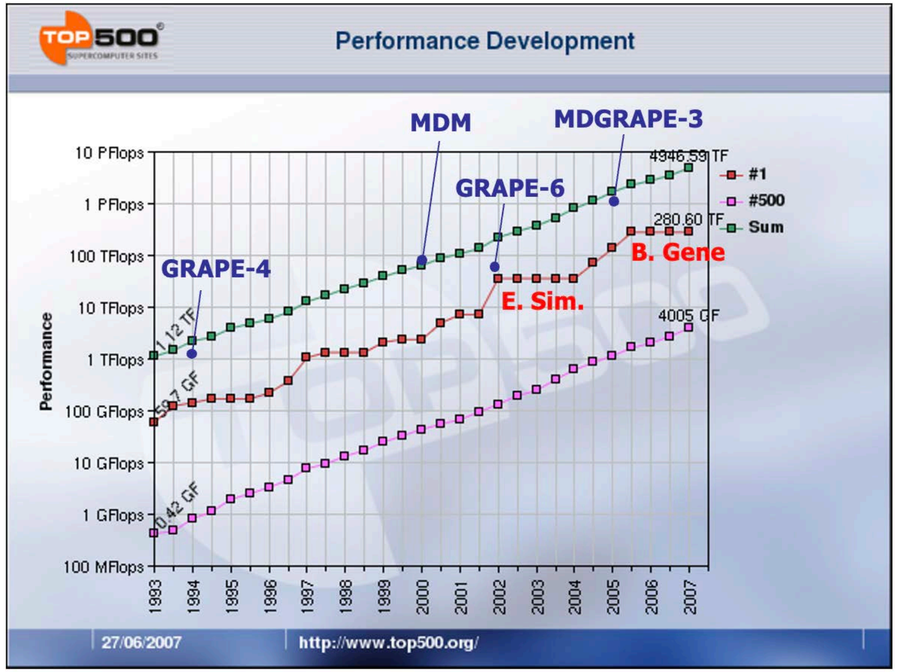


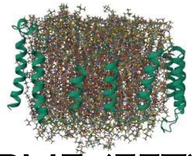
I want N-body

MDGRAPE-3

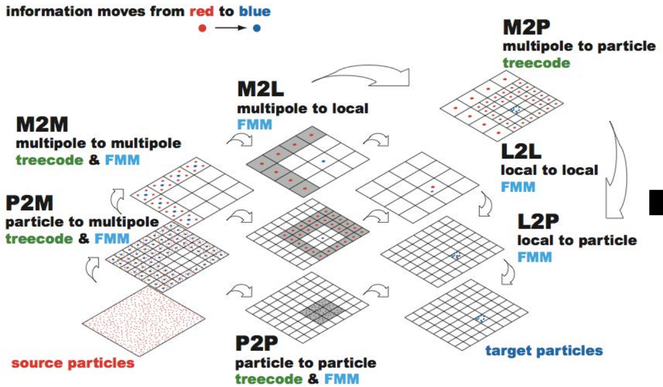
$$F = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}^2}$$

Hardware accelerator for N-body kernel

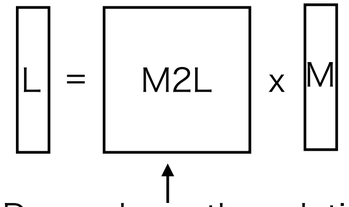




PME (FFT)



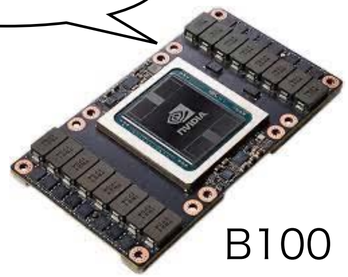
Fast Multipole Method



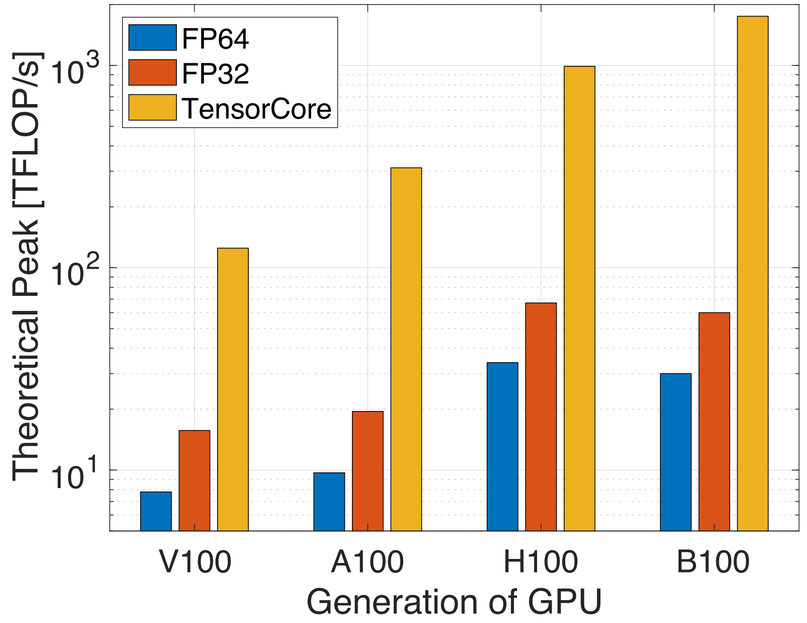
Depends on the relative position of cells

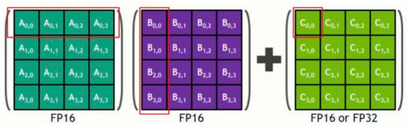
GEMM

I want GEMM



B100





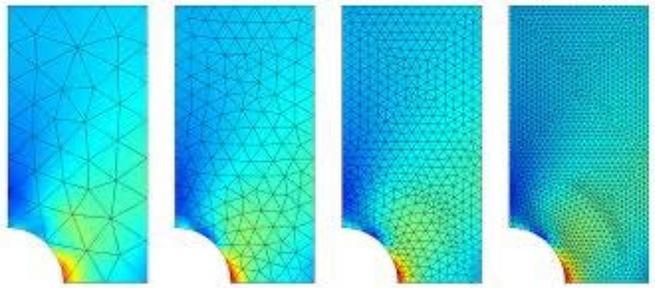
Hardware accelerator for Low-precision GEMM

True Ingenuity Lies in These Simple Abstractions

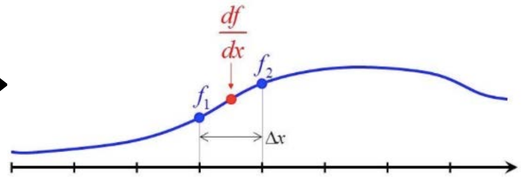
Mesh

Piece-wise linear functions

$$\frac{\partial f}{\partial x} = \frac{f_2 - f_1}{\Delta x}$$



https://www.comsol.jp/multiphysics/mesh-refinement



$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

Millennium Prize Problems

- Birch and Swinnerton-Dyer conjecture
- Hodge conjecture
- Navier–Stokes existence and smoothness
- P versus NP problem
- Poincaré conjecture (solved)
- Riemann hypothesis
- Yang–Mills existence and mass gap

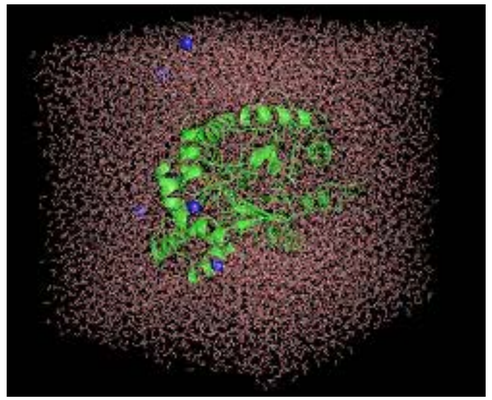
$$Ax = b$$

Basic linear algebra
Clear cut API

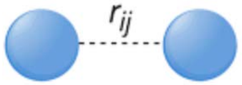
Particle

Pair-wise linear functions

$$F = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}^2}$$



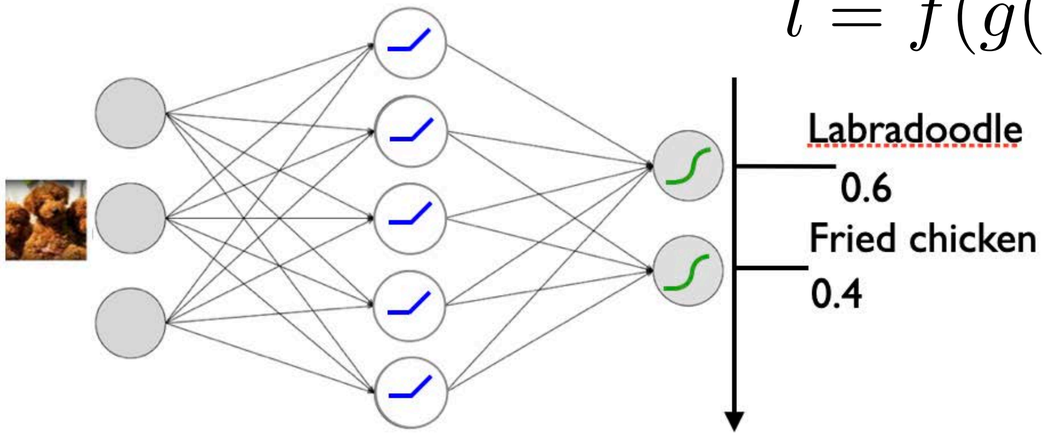
https://qstatix.co.in/molecular-dynamics-simulations/



Neural Network

Composite (mostly) linear functions

$$l = f(g(h(x)))$$



Forward propagation

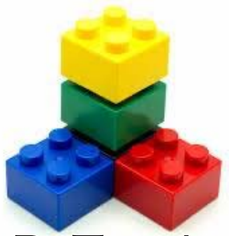
$$x_0 \quad x_1 = ReLU(z_0) \quad p = softmax(z_1)$$

$$z_0 = W_0 x_0 \quad z_1 = W_1 x_1 \quad l = -\log p$$

Backward propagation

$$\frac{\partial x_1}{\partial z_0} * \frac{\partial z_1}{\partial x_1} * \frac{\partial l}{\partial z_1} = \frac{\partial l}{\partial \theta}$$

$$\frac{\partial z_0}{\partial W_0} \quad \frac{\partial z_1}{\partial W_1}$$



PyTorch

Thanks to all my Collaborators