Yaroslav

Software is slow to adapt to changes of hardware Sutro meeting #11 (energy) Agenda

(dSuboptimization)
Cul-de-sacs (designed for horses)

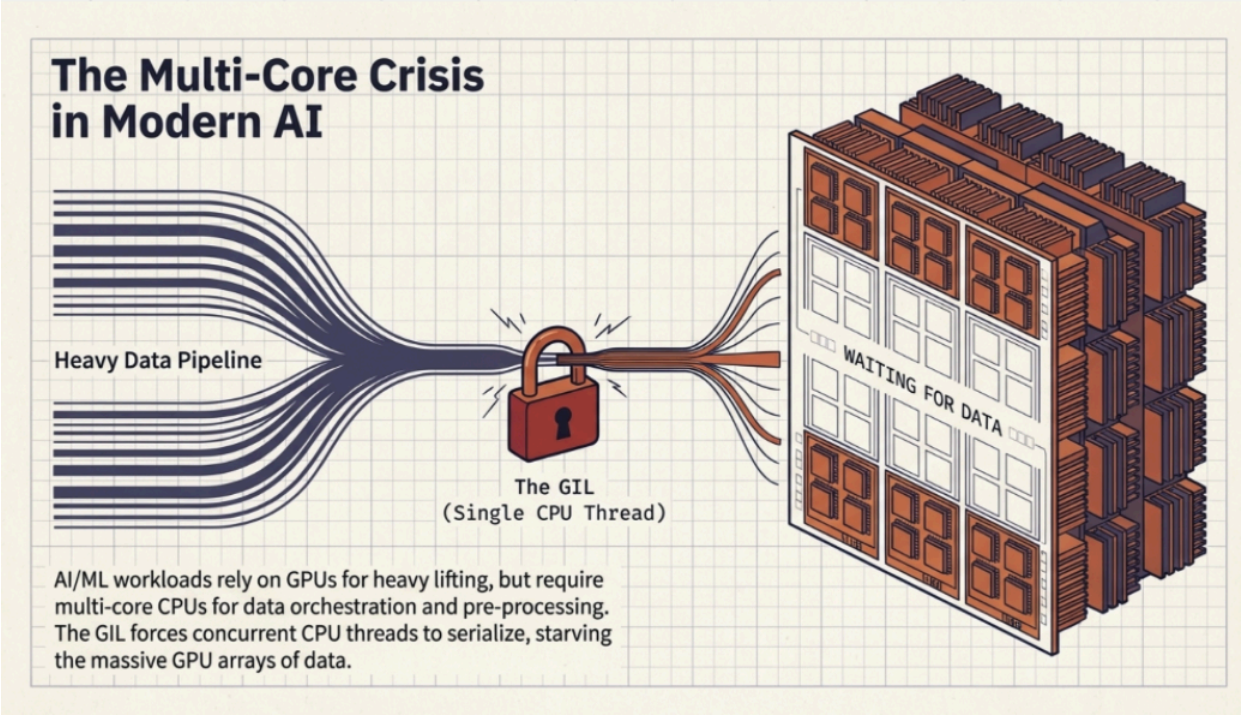


Lightwater reactors (designed for submarines)

were too focused on the old component (B-Tree) to see the system change.

- **Nuclear Power: The "Submarine" Trap**
  - **The Lock-in:** Light Water Reactors (LWR) are the standard for nuclear power, but they are not the safest or most efficient design. They became the standard because **Admiral Rickover** needed a compact reactor for *submarines*.
  - **The Failure:** The civilian industry just copied the submarine supply chain. We spent 50 years optimizing a "submarine engine" for cities, ignoring better designs (like Molten Salt Reactors) because the supply chain (the "interface") already existed.

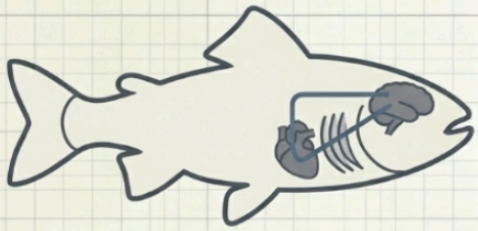
Python GIL (designed for single-core)



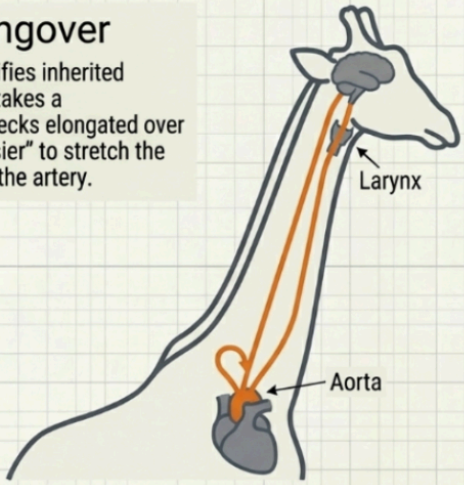
Recurrent laryngeal nerve chat
- Embryonic morphogenesis program.
- vs "the body geometry"

Biology: The Evolutionary Hangover

Natural selection cannot design from scratch; it modifies inherited structures. The recurrent laryngeal nerve in giraffes takes a preposterous 15-foot detour because, as vertebrate necks elongated over millions of years, incremental mutations made it "easier" to stretch the nerve than to structurally sever and reroute it across the artery.

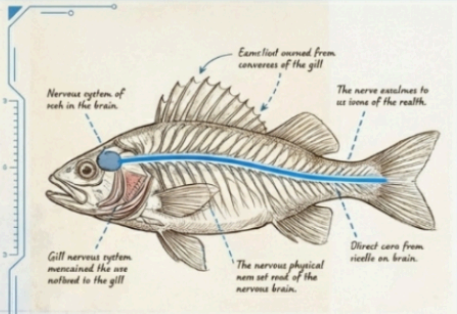


Fish: Direct neural path.

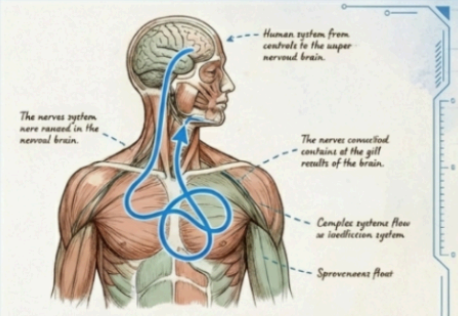


Giraffe: 15-foot path-dependent detour.

Biological Tech Debt



Simple System (Direct Path)

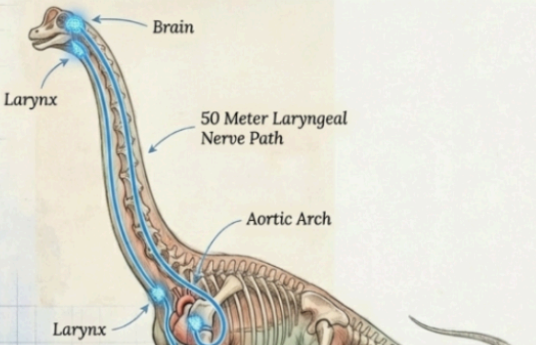


Complex System (Tangled Loop)

The Premise:
When complex systems evolve from simple ones without a top-down redesign, early bugs become permanent architectural constraints.

The Context:
A biology lecture delivered to crypto-engineers to explain systemic bottlenecks and network inefficiencies.

The Ultimate Bottleneck: The Sauropod Nerve



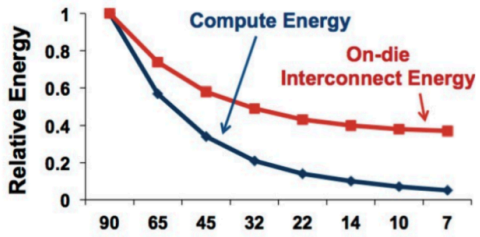
The Flaw
The longest single cell in the history of life.

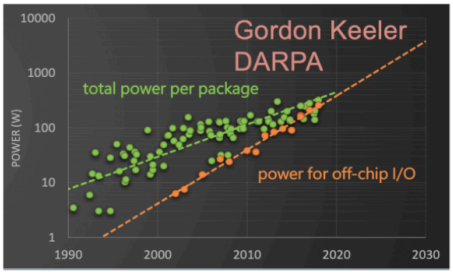
The Origin
In early fish, this nerve ran a few inches straight from the brain to the gills. As necks lengthened over millions of years, the nerve was already trapped under the heart's aortic arch.

The Result

Hardware has changed.

1. the end of Dennard scaling
    one fast core ==> many slow cores
2. expensive arithmetic => expensive memory





memory wall chat

Design decisions

- Gradient descent
    Black box optimizers
        then: no market fit => shifting demands
        now: next character prediction is worth a lot
- Layers
    Sequential
        then: single core
        now: 132 SMs of Hopper can can run different instructions
- Backprop
    Low compute/commute ratio:
        then: fine because no memory wall
        now: memory wall
    literally have to drop and recompute same quantities over and over (checkpointing)
- Synchronous training
        then: small number of devices
        now: many devices, things fail

Past: Low-level => high-level (Dennard scaling)
Present: high-level primitives unclear

1. We're going lower level chat

    - Matlab (pre 2012)
    - PyTorch/TENSORFLOW
    - CUDA
    - PTX?

    Compiler designers are trying to keep development abstracted, mixed success.
    https://www.modular.com/blog/democratizing-compute-part-1-deepseeks-impact-on-ai

    DeepSeek used raw PTX + infiniband primitives
    DSL proliferation

    Timeline view

    2021–2023: Triton became the productivity default for custom ML kernels; CUDA/CUTLASS remained the expert-performance default.

    2024: Hopper/H100 exposed the limits of first-generation Triton for peak attention. FlashAttention-3 used CUTLASS/CuTe primitives, WGMMA, TMA, warp specialization, and FP8. GPU Mode's public materials around this period include CUTLASS, FlashAttention-3, FlashInfer, Mosaic GPU, and CuTe. arXiv +2

    2025: Tile DSLs proliferated. Tilus, TileLang, ThunderKittens, Helion, Gluon/TLX, Cypress, Tawa, DeepGEMM, and Mirage/MPK all fit the same pattern: more specialization, more explicit scheduling, and/or more compiler-managed orchestration. GitHub +4

    2026: Blackwell/B200 made the trend unavoidable. FlashAttention-4 is CuTe DSL; CUDA Tile/TileIR becomes a visible NVIDIA compiler substrate; DeepSeek-V4 support uses TileLang kernels; ThunderKittens 2.0 adds Blackwell/MXFP8/NVFP4. arXiv +3

Changing blocks of the algorithm, but **usually not changing the algorithm**.
But algorithm design can't be decoupled from hardware

Exception mini-Batching

SGD => mini-batch SGD This is an example of adjusting the algorithm to fit the hardware. chat
    - main not have been intentional
https://www.sciencedirect.com/science/article/abs/pii/S0893608003001382
2003 paper -- minibatching is always worse

Abstract

Gradient descent training of neural networks can be done in either a *batch* or *on-line* manner. A widely held myth in the neural network community is that batch training is as fast or faster and/or more 'correct' than on-line training because it supposedly uses a better approximation of the true gradient for its weight updates. This paper explains why batch training is almost always slower than on-line training—often orders of magnitude slower—especially on large training sets. The main reason is due to the ability of on-line training to follow curves in the error surface throughout each epoch, which allows it to safely use a larger learning rate and thus converge with less iterations through the training data. Empirical results on a large (20,000-instance) speech recognition task and on 26 other learning tasks demonstrate that convergence can be reached significantly faster using on-line training than batch training, with no apparent difference in accuracy.

2006 -- "A Fast Learning Algorithm for Deep Belief Nets"
https://www.cs.toronto.edu/~fritz/absps/ncfast.pdf

> work was 1.25% errors on the official test set. The network shown in Figure 1 was trained on 44,000 of the training images that were divided into 440 balanced mini-batches, each containing 10 examples of each digit class.[6] The weights were updated after each mini-batch.

Main cost today is moving bits around.

> Batching saves on data movement.

Jeff Dean -- It may cost a thousand picajoules to get parameters from one side of the chip to another,

> Once you brought it over, you might as well use it several times.
> https://www.latent.space/p/jeffdean

Goal

Go beyond "energy-efficient kernel" to energy-efficient **algorithms**

Approach: series of challenges that focus on reduced Joules
Yaroslav's sutro planning sprint #1

Competitions

Theory track:
idealized Joules

> Joules mostly come from communication, reduce mm's of communication

Component	Energy	What it represents
On-chip communication	**~100 fJ/bit-mm**	Moving one bit one millimeter across on-chip wires
Small SRAM access	**~50 fJ/bit**	Reading/writing one bit from a small (local) SRAM macro
Arithmetic (add)	**~1 fJ/bit**	One bit of addition logic toggling

- Bill Dally Model

- Matmul challenge
- Sparse parity challenge
- (wip: subset of https://github.com/cybertronai/hinton-problems/)
- (wip: subset of https://github.com/cybertronai/schmidhuber-problems)

- tools
    - standalone researcher SutroAna
    - Telegram+docs-integrated researcher SutroYaro

Applied track -- actual Joules on GPUs

- wikitext