# High-Performance Computing for Scientific Discovery

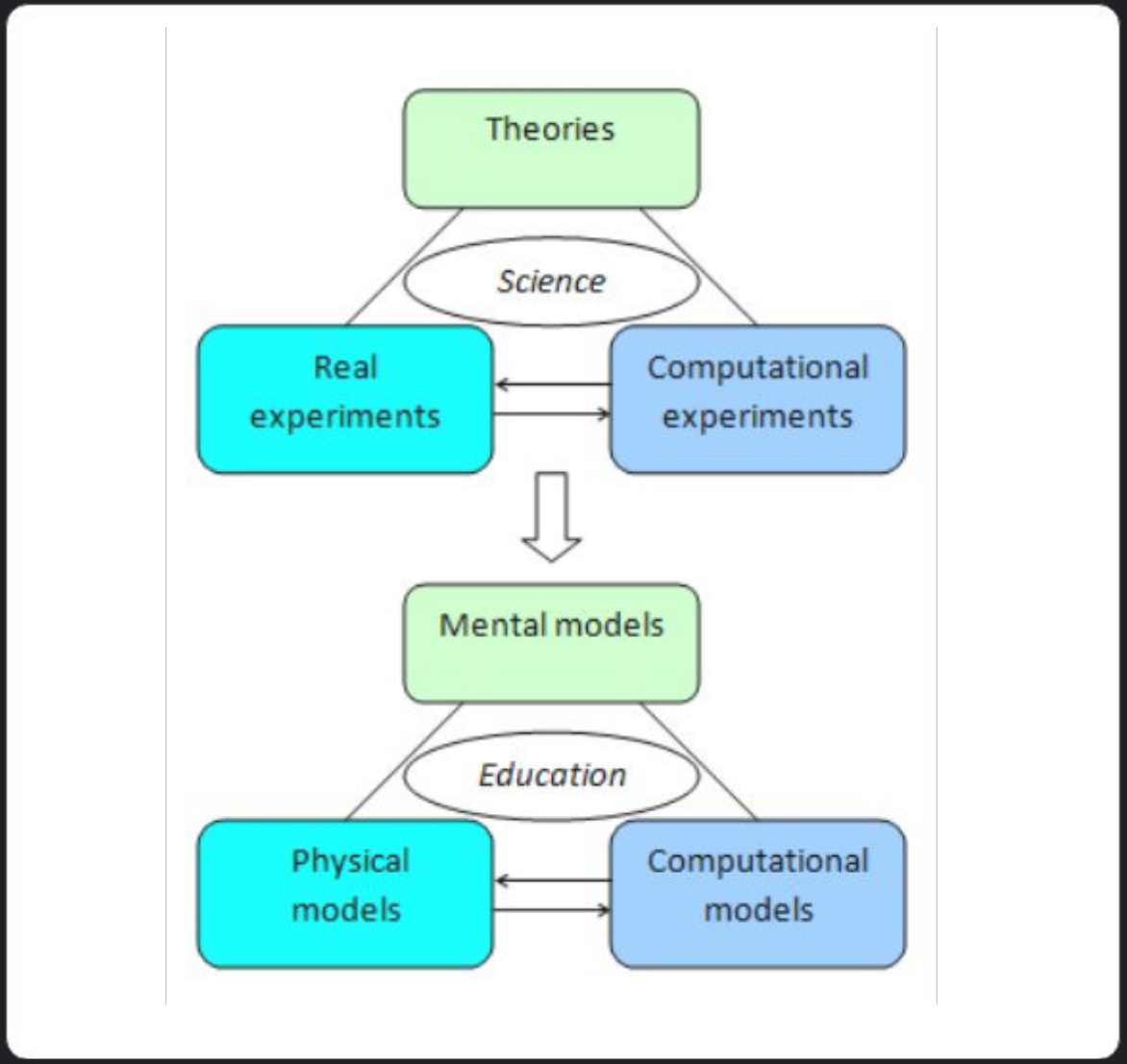
## Looking Under the Hood: Software, TPUs, and Keras 3.0

# HPC: The Third Pillar of Science

Historically, science stood on two pillars: **Theory** and **Experimentation**.

Today, **Simulation (Computation)** is the indispensable third pillar, bridging the gap when theory is too complex and experiments are too dangerous or impossible.

This triad, as shown in the diagram, now forms the foundation of modern discovery.



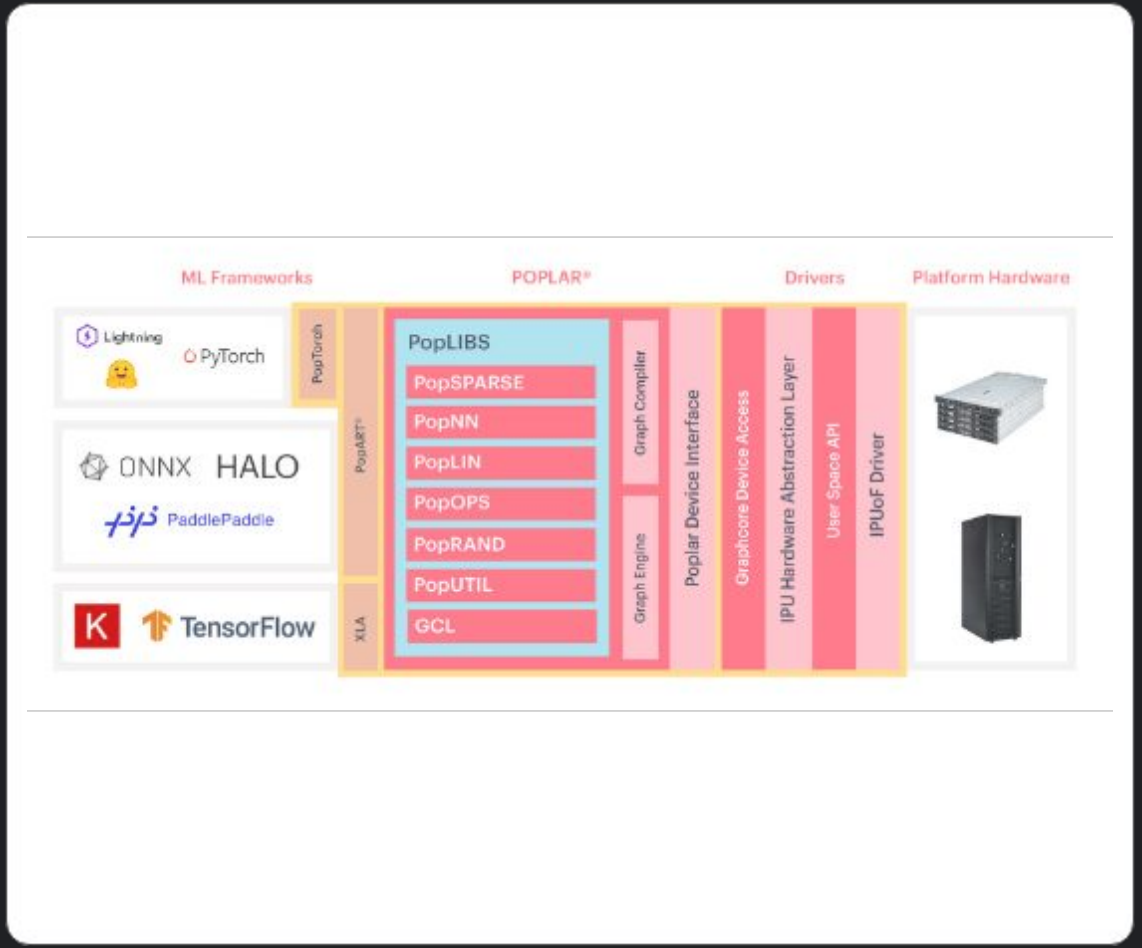
# The Data Bottleneck


- As this graph illustrates, genomic data generation is outpacing standard hardware improvements (Moore's Law) by a massive margin.
- While compute power roughly doubles every two years, biological data is doubling much faster (every 6-12 months).
- This widening gap means we cannot just "wait for faster chips"; we must optimize software.

# The Google Software Bridge

This diagram illustrates the comprehensive stack bridging biological applications with Google Cloud TPUs.

- **ML Frameworks (Top Layer):** Keras 3.0 provides the unified API, allowing biologists to use JAX, TensorFlow, or PyTorch backends seamlessly.
- **Compiler Engine (Middle Layer):** XLA (Accelerated Linear Algebra) ingests computational graphs from these frameworks, performing hardware-agnostic optimizations like generic operator fusion.
- **Hardware Abstraction (Lower Layer):** Specialized JIT compilers translate XLA graphs into machine code tailored for the TPU systolic array, managing memory tiling and inter-chip interconnects.
- **Platform Hardware (Bottom Layer):** The stack runs on Cloud TPU v5 Pods designed for exascale matrix math.

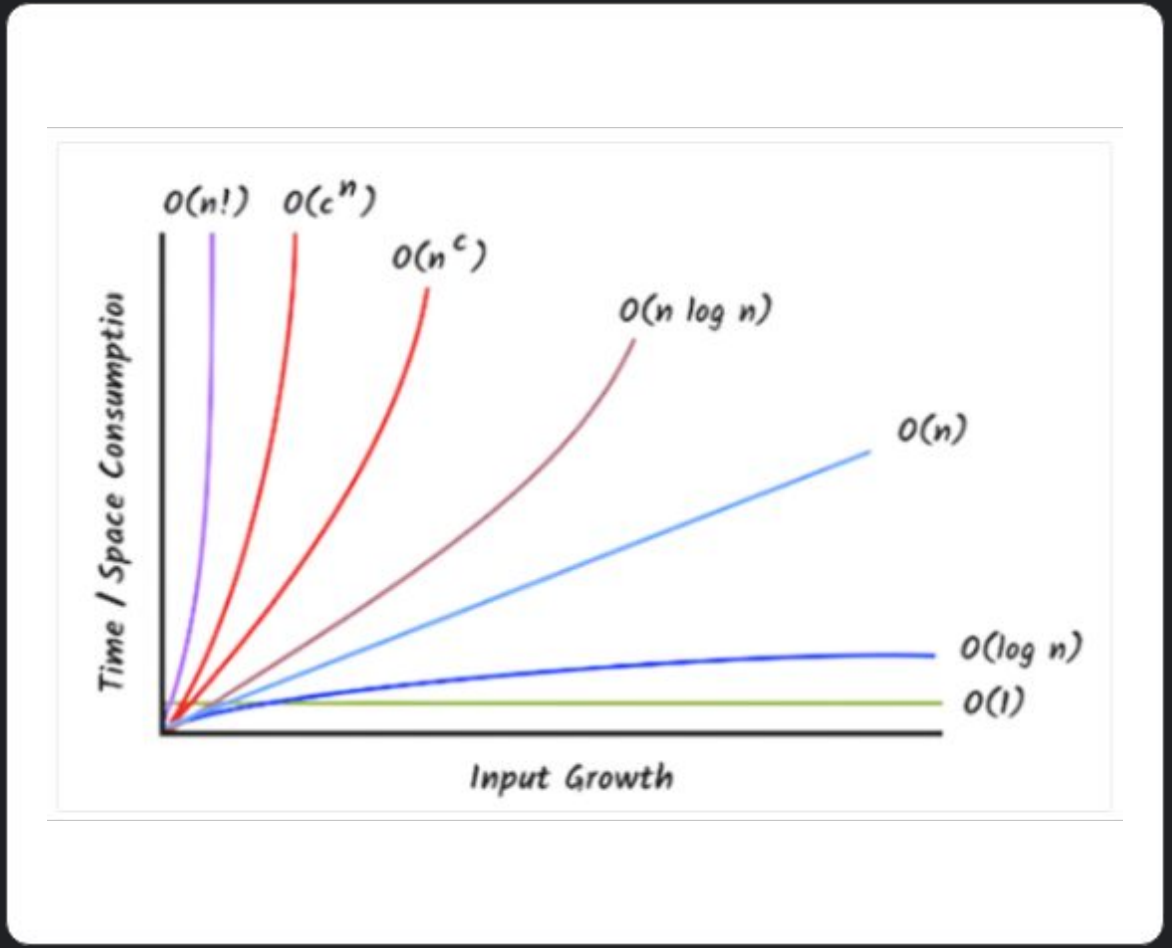


# Section 1: The Challenge

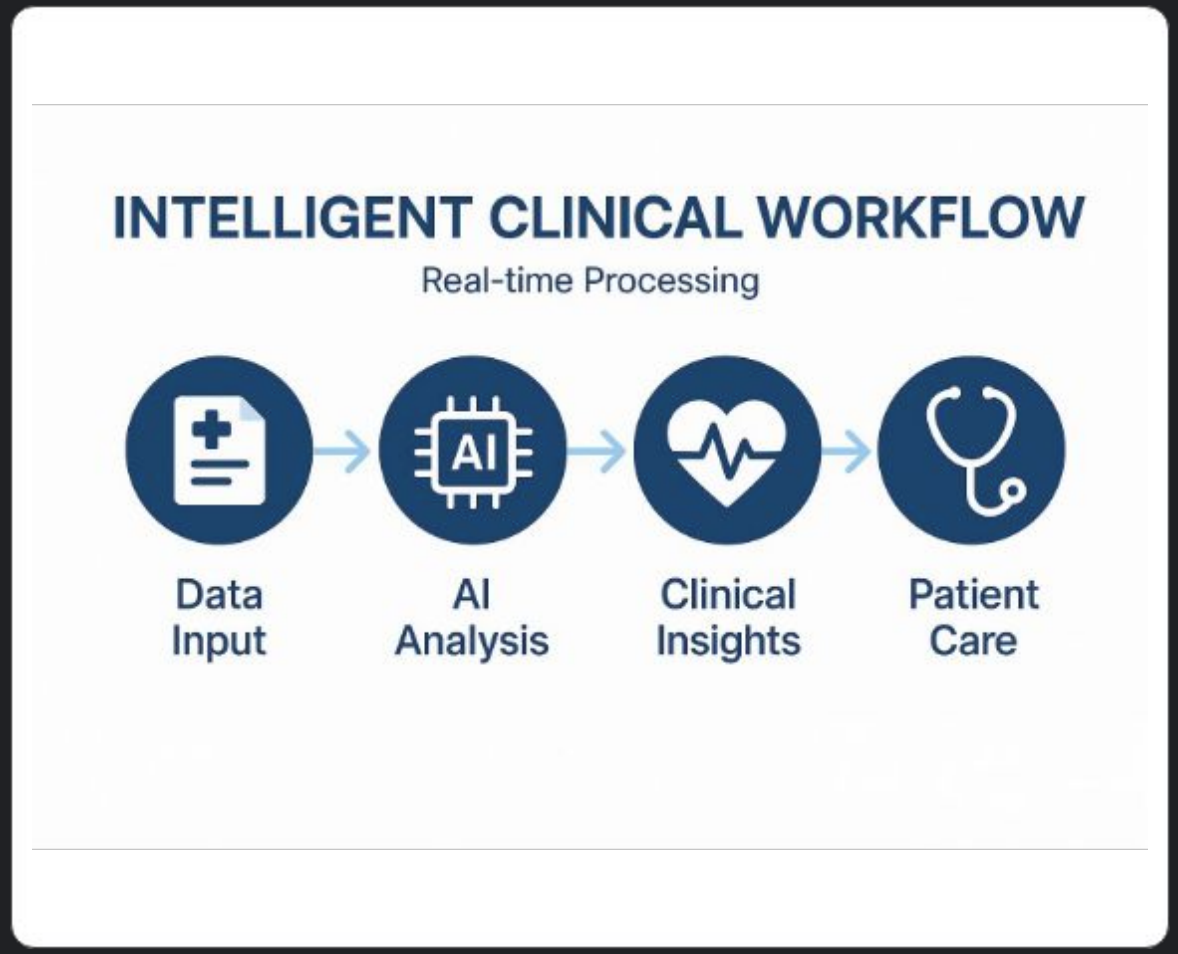
Unique Workloads in Life Sciences

# Genomics: The quadratic problem

- Standard Transformer models have **quadratic complexity $O(N^2)$**.
- As seen in the graph, memory usage (y-axis) explodes exponentially as the DNA sequence length (x-axis) increases.
- A 100k base-pair sequence is impossible on standard hardware without specialized optimization.



# Clinical Trials: The Latency Problem




In this clinical workflow, the "AI Inference" box is on the critical path.

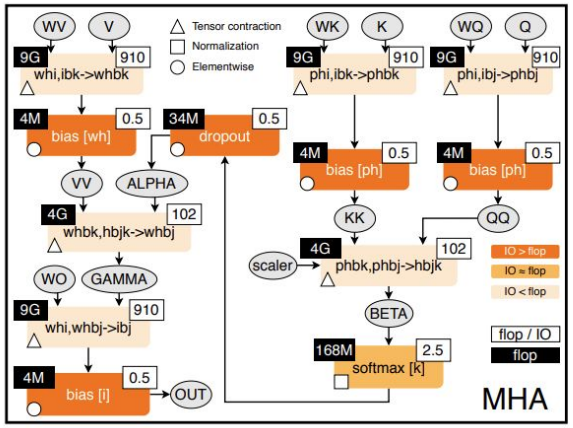
A doctor waiting for a patient match recommendation cannot tolerate a 30-second delay.

Optimizing this single step directly impacts patient care velocity.

# The Von Neumann Bottleneck

> "Throwing more hardware at the problem doesn't work if your software spends 80% of its time moving data instead of computing."

```python
@dace.program
def mha_forward(
  wq: dace.float16[P, H, I], bq: dace.float16[P, H],
  q: dace.float16[I, B, J],
  wk: dace.float16[P, H, I], bk: dace.float16[P, H],
  k: dace.float16[I, B, K],
  wv: dace.float16[W, H, I], bv: dace.float16[W, H],
  v: dace.float16[I, B, K],
  wo: dace.float16[W, H, I], bo: dace.float16[I],
  scaler: dace.float16
):
  qq = np.einsum("phi,ibj->phbj", wq, q) + bq[:,:,None,None]
  kk = np.einsum("phi,ibk->phbk", wk, k) + bk[:,:,None,None]
  vv = np.einsum("whi,ibk->whbk", wv, v) + bv[:,:,None,None]
  beta = scaler * np.einsum("phbk,phbj->hbjk", kk, qq)
  alpha = dropout(softmax(beta))
  gamma = np.einsum("whbk,hbjk->whbj", vv, alpha)
  out = np.einsum("whi,whbj->ibj", wo, gamma)+bo[:,None,None]
  return out
```




Data must travel across the limited bus (red arrows) between memory and compute units.

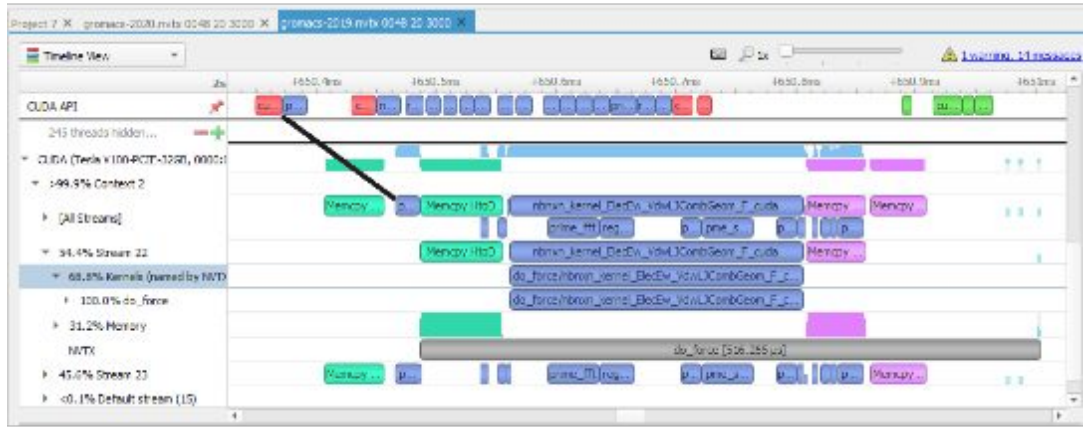
# Section 2: Under the Hood

Compilers & XLA

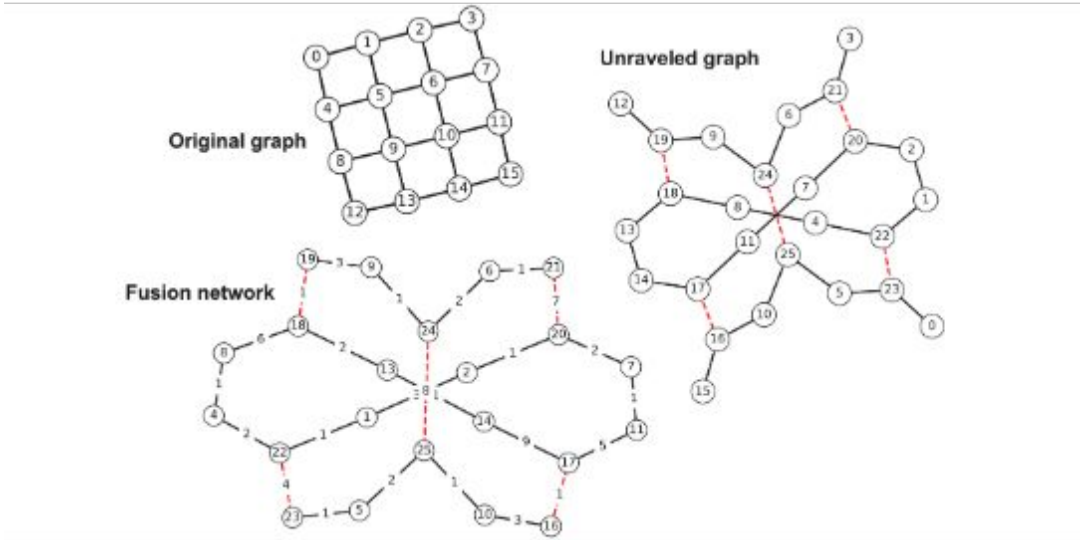
# The Invisible Overhead

This timeline shows GPU activity. Notice the **gaps** between the colored blocks?

That's "dead air"—overhead from launching thousands of tiny individual kernels. The GPU is idle during these times, waiting for instructions.



# The Mechanics of Operator Fusion




Deep learning accelerators have a massive gap between compute speed (FLOPS) and memory bandwidth.

**The Problem:** Standard execution launches individual 'kernels' for every math operation (add, multiply, activate), forcing data to travel back and forth to slow High-Bandwidth Memory (HBM).

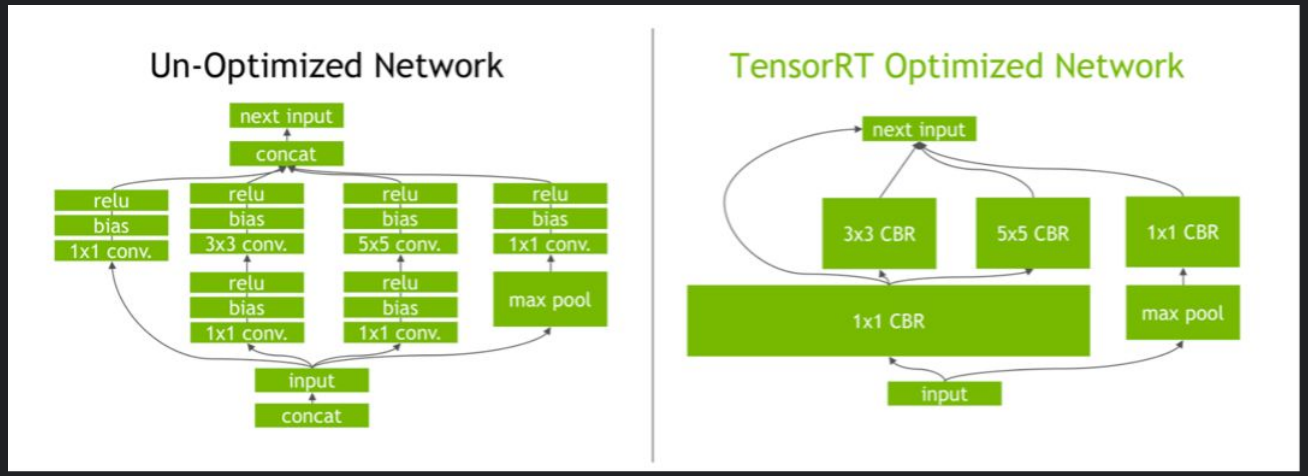
**The Fix:** Fusion analyzes the computational graph to identify sequences of element-wise operations (like `Ax + b` followed by `ReLU`).

**The Result:** It compiles these into a single hardware kernel, keeping intermediate data in ultra-fast on-chip SRAM and bypassing HBM entirely.

# Visualizing the Memory Wall

This diagram compares two execution modes for a standard deep learning block:

- **Without Fusion (Top):** Operations like Conv2D write massive intermediate outputs entirely to external DDR/HBM memory. Subsequent operations (like ReLU) must slowly read it all back. This "ping-pong" effect kills performance.
- **With Fusion (Bottom):** The data never leaves the chip between operations. It stays in fast local SRAM (on-chip memory). We only pay the expensive "DDR Tax" once at the very end.



# Deep Dive: Google Cloud XLA

XLA (Accelerated Linear Algebra) is the domain-specific compiler that unlocks TPU performance, going beyond simple fusion:

- **Graph Optimization:** Performs high-level passes like dead code elimination and constant folding before compilation even begins.
- **Buffer Assignment:** Intelligently reuses on-chip memory buffers to minimize the active memory footprint during training.
- **TPU-Specific Code Gen:** Generates VLIW (Very Long Instruction Word) machine code specifically tuned to keep the TPU's massive matrix units fed with data without stalling.

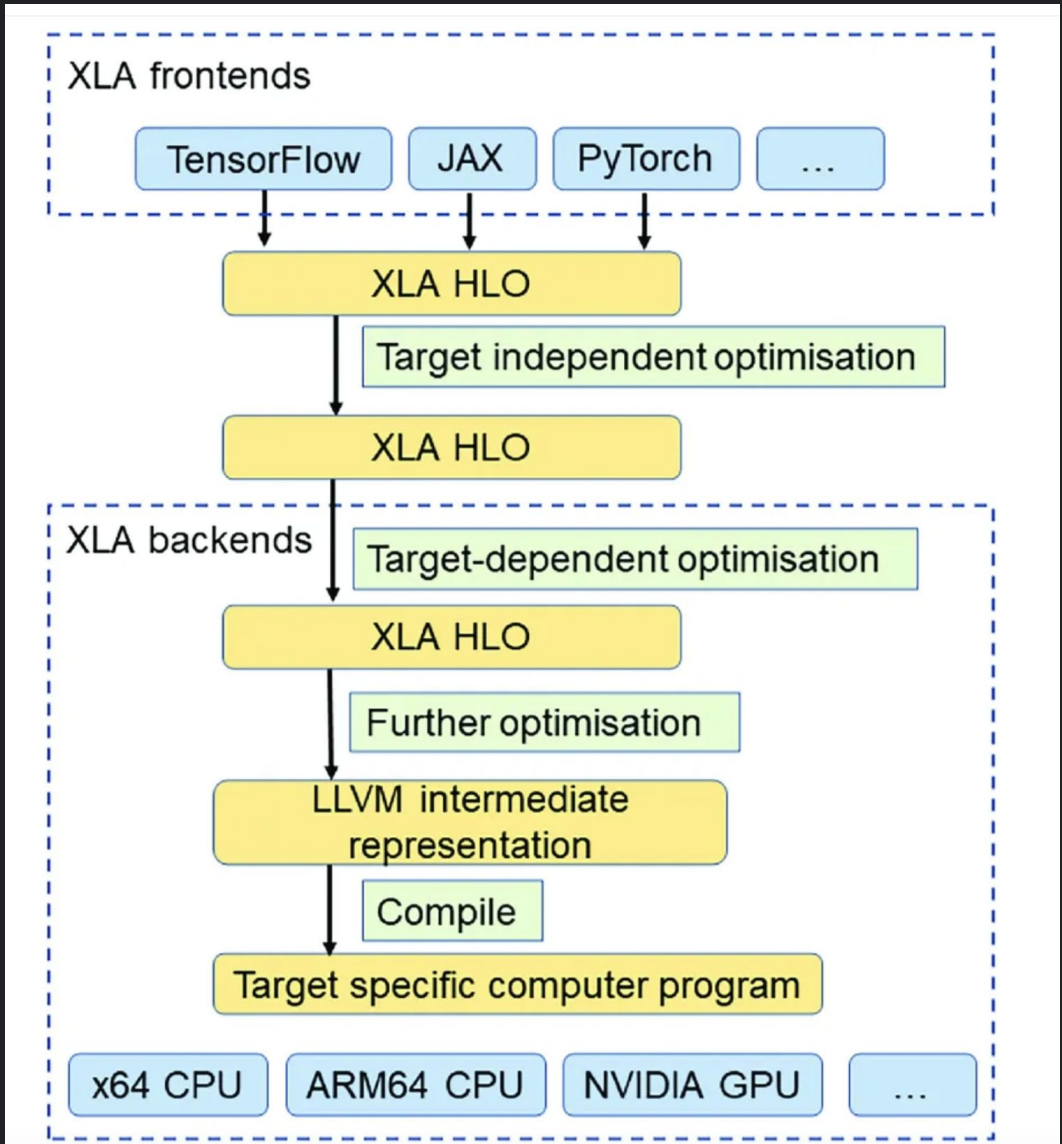



Figure 5.1 Architecture of accelerated linear algebra (XLA) compiler.

# Section 3: Precision & Quantization

Doing More with Less

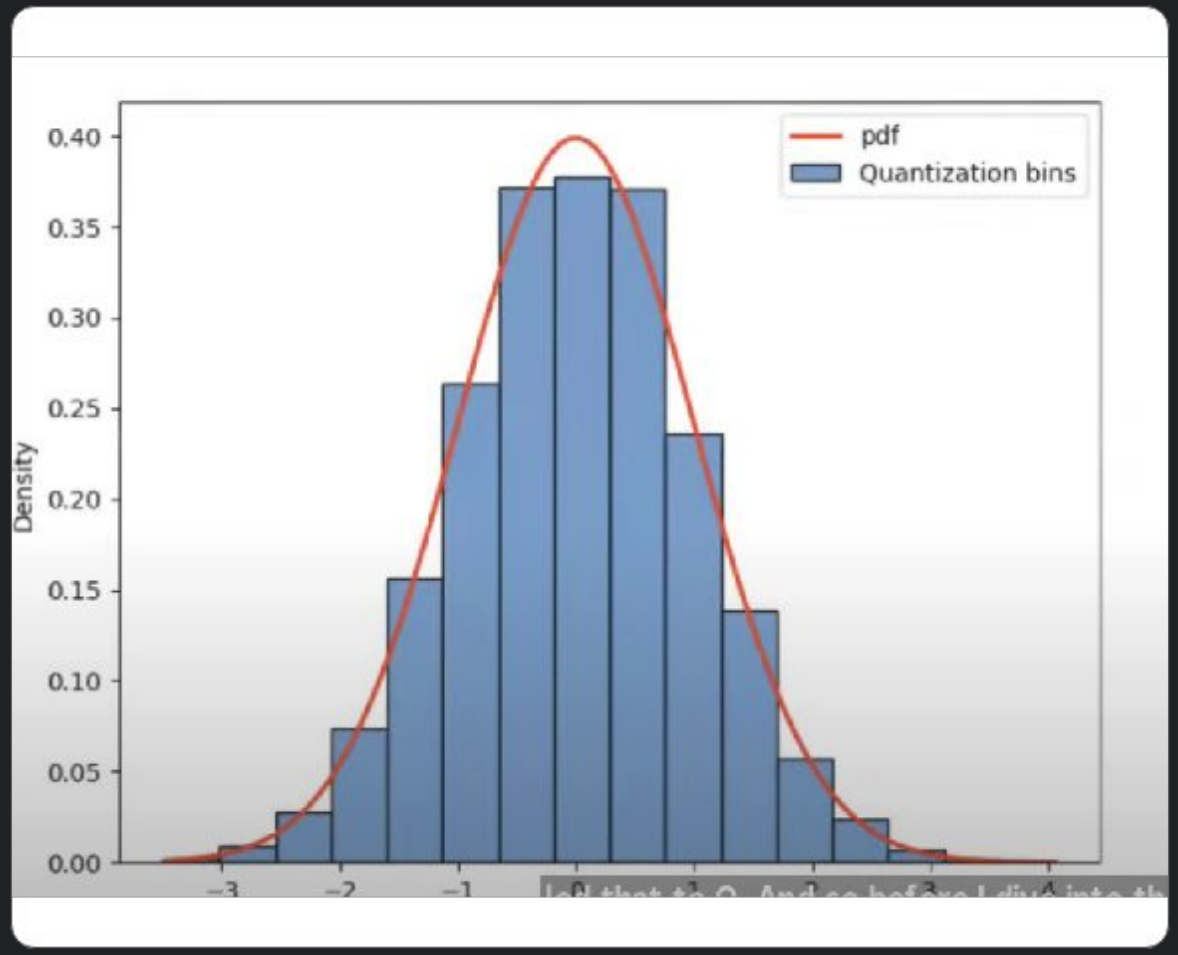
Memory

# The Hard Limits of FP32 Precision

32-bit Full Precision (FP32) has long been the scientific standard, but it hits a hard wall with modern Foundation Models.

- **The Math:** A 70B parameter Llama-based bio-model requires 4 bytes per parameter. That is ~280GB just to load the weights, before any training data.
- **The Reality:** A single commercial accelerator typically has between 16GB to 80GB of High-Bandwidth Memory (HBM).
- **The Consequence:** You literally cannot load these models on standard hardware without complex, slow model-parallelism across dozens of chips.

# The NF4 Breakthrough



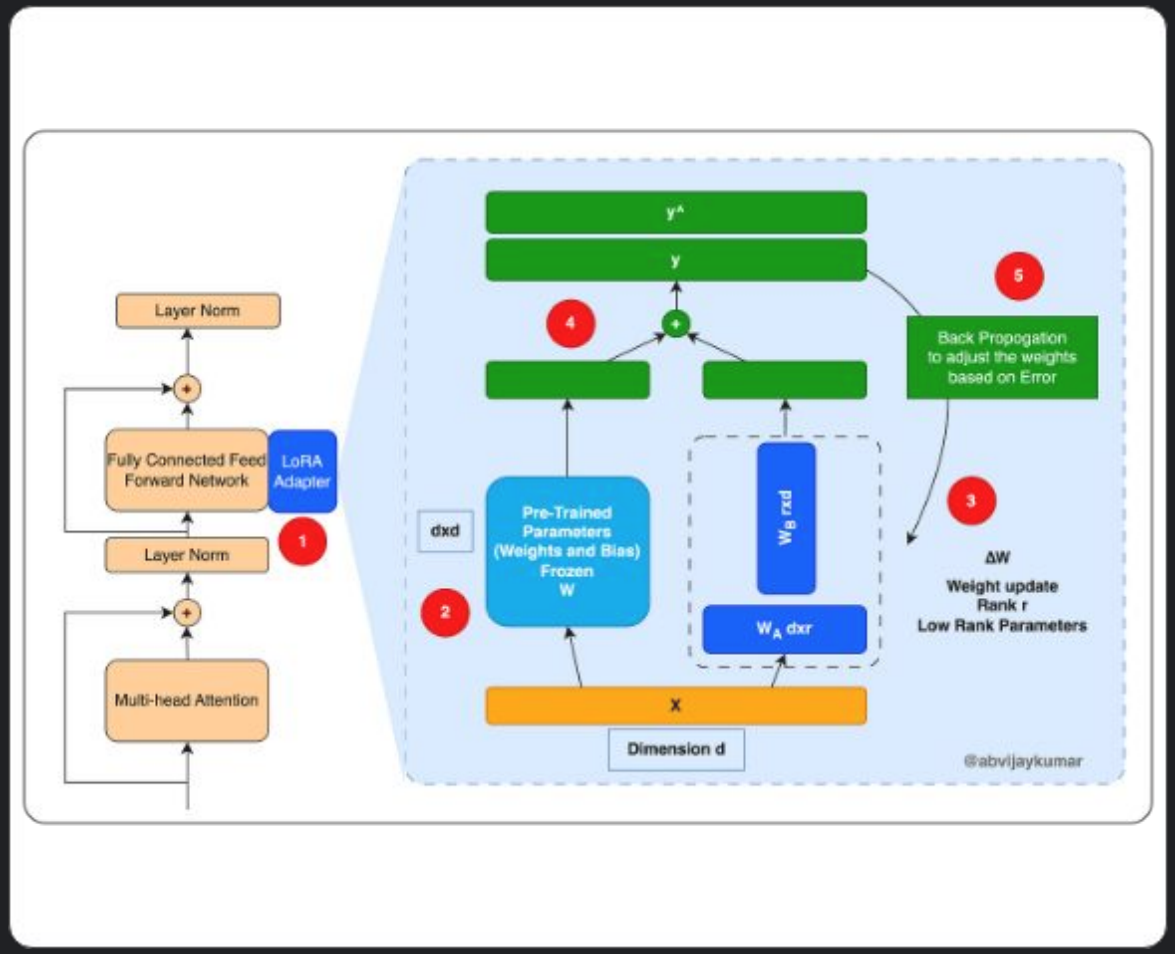

4-bit Normal Float (NF4) is an information-theoretically optimal data type for this problem.

- **Bio-Weights are Normal:** As shown in the bell curve diagram, pre-trained neural network weights naturally cluster around zero.
- **Smart Allocation:** Unlike standard uniform quantization (which wastes bits on empty ranges), NF4 allocates more representational values near the center of the distribution where most weights live.
- **Result:** We can store that 280GB model in just ~35GB with virtually no loss in reasoning ability.

# HPC Application: QLoRA

**Quantized Low-Rank Adaptation (QLoRA)** allows us to fine-tune massive models on a single GPU/TPU.

- Freeze the massive base model in 4-bit (NF4).
- Train only small "adapter" layers in higher precision.



# No Free Lunch? Actually, Yes.

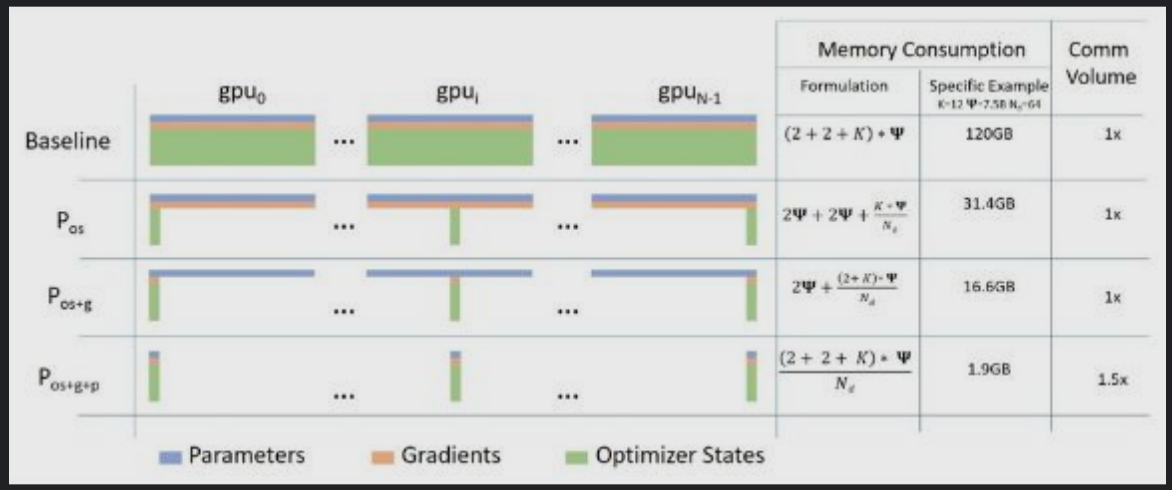
| | Mean 5-shot MMLU Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| LLaMA Size | 7B | | 13B | | 33B | | 65B | | Mean |
| Dataset | Alpaca | FLAN v2 | Alpaca | FLAN v2 | Alpaca | FLAN v2 | Alpaca | FLAN v2 | |
| BFloat16 | 38.4 | 45.6 | 47.2 | 50.6 | 57.7 | 60.5 | 61.8 | 62.5 | 53.0 |
| Float4 | 37.2 | 44.0 | 47.3 | 50.0 | 55.9 | 58.5 | 61.3 | 63.3 | 52.2 |
| NFloat4 + DQ | 39.0 | 44.5 | 47.5 | 50.7 | 57.3 | 59.2 | 61.8 | 63.9 | 53.1 |

- Quantization Maintains Performance: The most significant finding is that the 4-bit quantized models (Float4 and NFloat4 + DQ) achieve accuracy very similar to the 16-bit baseline (BFloat16). This indicates that you can significantly reduce the memory footprint of these models without a major loss in capabilities.
- NFloat4 + DQ Efficiency: The NFloat4 + DQ method often performs slightly better than standard Float4 and sometimes even matches or slightly exceeds the BFloat16 baseline (e.g., at 65B FLAN v2, it scores 63.9 vs the baseline 62.5).

# Scaling Up: DeepSpeed ZeRO

When models are too big even for quantization, we must shard them across many devices.

**ZeRO Stage 1,2&3** automatically partitions model parameters, gradients, and optimizer states across all available TPUs/GPUs, acting as one giant accelerator.




| | Memory Consumption | | Comm Volume |
|---|---|---|---|
| | Formulation | Specific Example | |
| Baseline | $(2+2+K)*\Psi$ | 120GB | 1x |
| $P_{os}$ | $2\Psi+2\Psi+\frac{K*\Psi}{N_d}$ | 31.4GB | 1x |
| $P_{os+g}$ | $2\Psi+\frac{(2+K)*\Psi}{N_d}$ | 16.6GB | 1x |
| $P_{os+g+p}$ | $\frac{(2+2+K)*\Psi}{N_d}$ | 1.9GB | 1.5x |

ZeRO (Zero Redundancy Optimizer) reduces GPU memory by partitioning model data rather than replicating it. It operates in three stages, successively sharding optimizer states (Stage 1), gradients (Stage 2), and model parameters (Stage 3). This drastically lowers per-GPU memory consumption, e.g., from 120GB to 1.9GB.

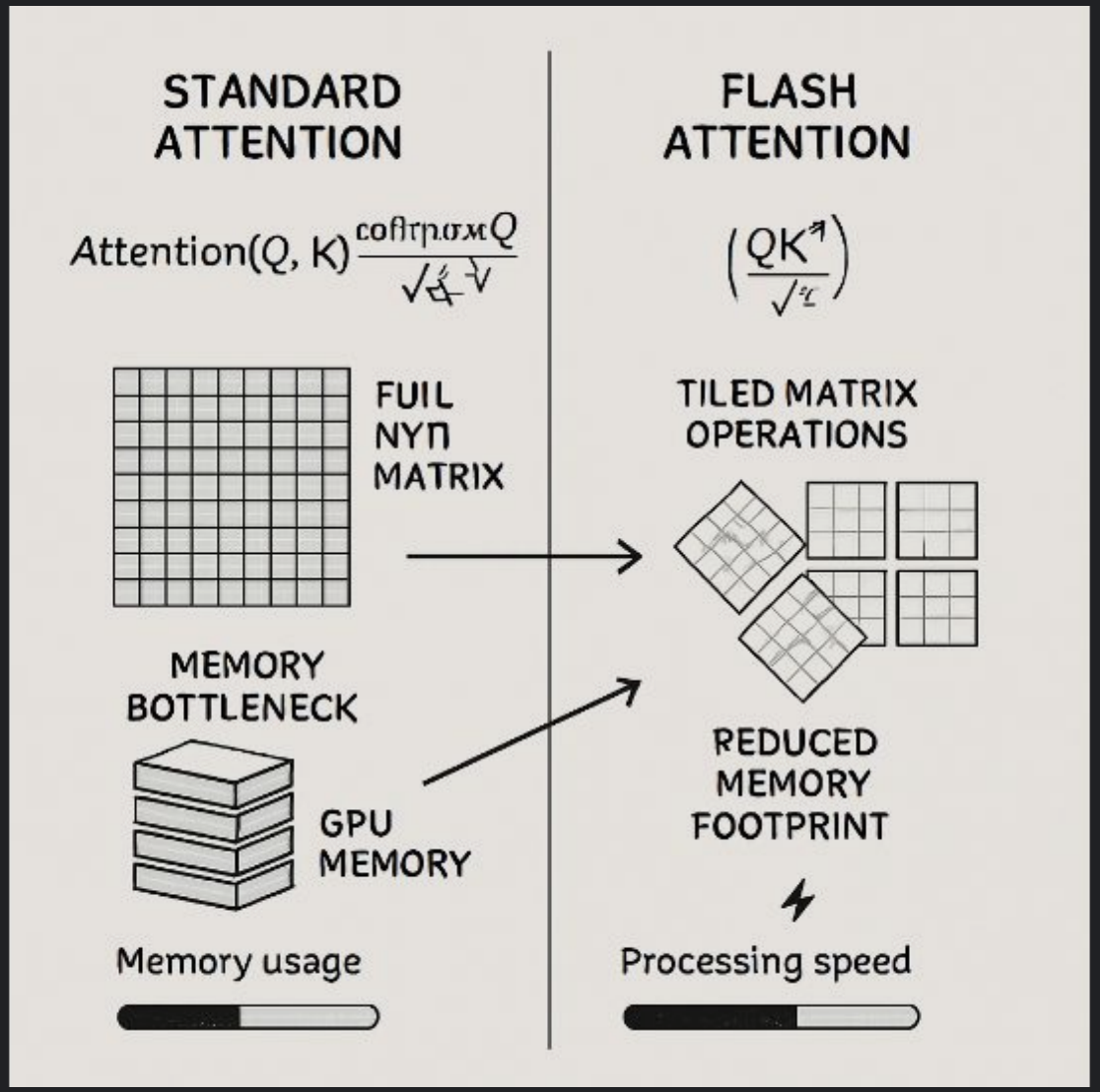
# Section 4: Optimization Frameworks

Taming Transformers & Democratizing AI

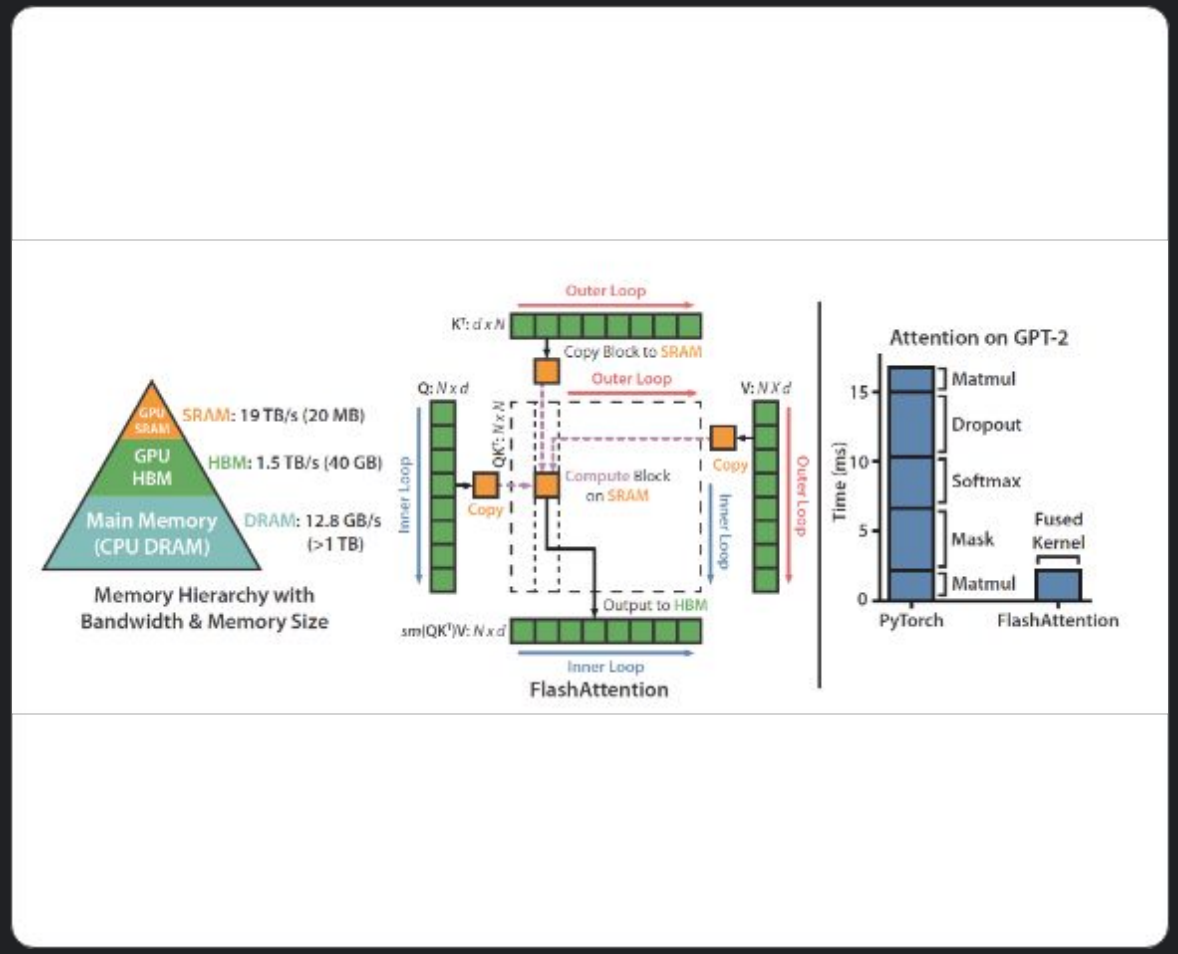
# The Attention Bottleneck

Transformers (the basis of most modern bio-models) compare every token to every other token.

For a 100k base-pair genome sequence, this creates a massive 100k x 100k matrix that exceeds hardware memory.



# FlashAttention: IO-Awareness




This image explains how FlashAttention significantly speeds up standard attention mechanisms by addressing the "memory wall" problem.

1. The Problem: The Memory Wall: GPUs have fast but small SRAM (on-chip) and large but slow HBM (main memory). Standard attention constantly moves data between these, making data movement, not computation, the bottleneck.
2. The Solution: FlashAttention Tiling: FlashAttention is "IO-aware." It breaks the attention calculation into small blocks (tiles) that fit into fast SRAM. It loads small blocks of Query, Key, and Value matrices, computes the entire attention mechanism for that block within SRAM (fused computation), and only writes the final result to HBM, avoiding materializing the large attention matrix in slow memory.
3. The Result: Fused Kernel Speedup: A bar chart shows the impact on GPT-2 training. Standard PyTorch performs operations sequentially, requiring HBM access between each step, leading to longer times. FlashAttention fuses all steps into one SRAM-based kernel, drastically reducing memory access overhead and speeding up operations.

# Keras 3.0: The Unifying API

Write Once, Run Anywhere

Historically, you had to choose: easy (Keras/TF) or fast/flexible (JAX/PyTorch).

**Keras 3.0** changes this. It's a high-level API that can run on top of JAX, TensorFlow, or PyTorch seamlessly.

# Keras 3 for Life Sciences

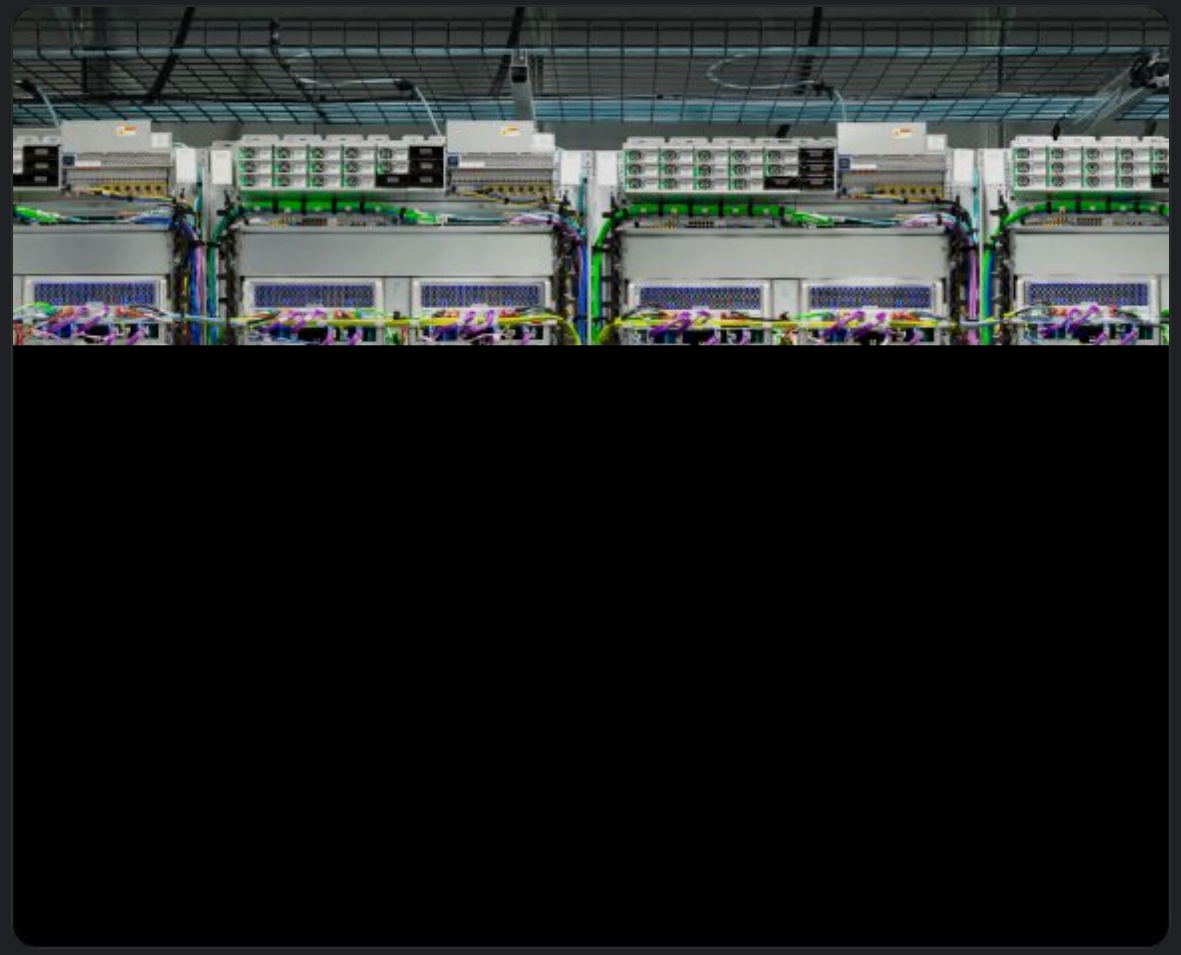
- **Portability:** Develop a clinical model on a laptop (using PyTorch backend), deploy it on a hospital Google Cloud TPU Pod (using JAX backend) without changing standard Keras code.
- **Performance:** Gives biologists access to JAX's extreme speed without needing to learn low-level functional programming.

# Section 5: Specialized Hardware

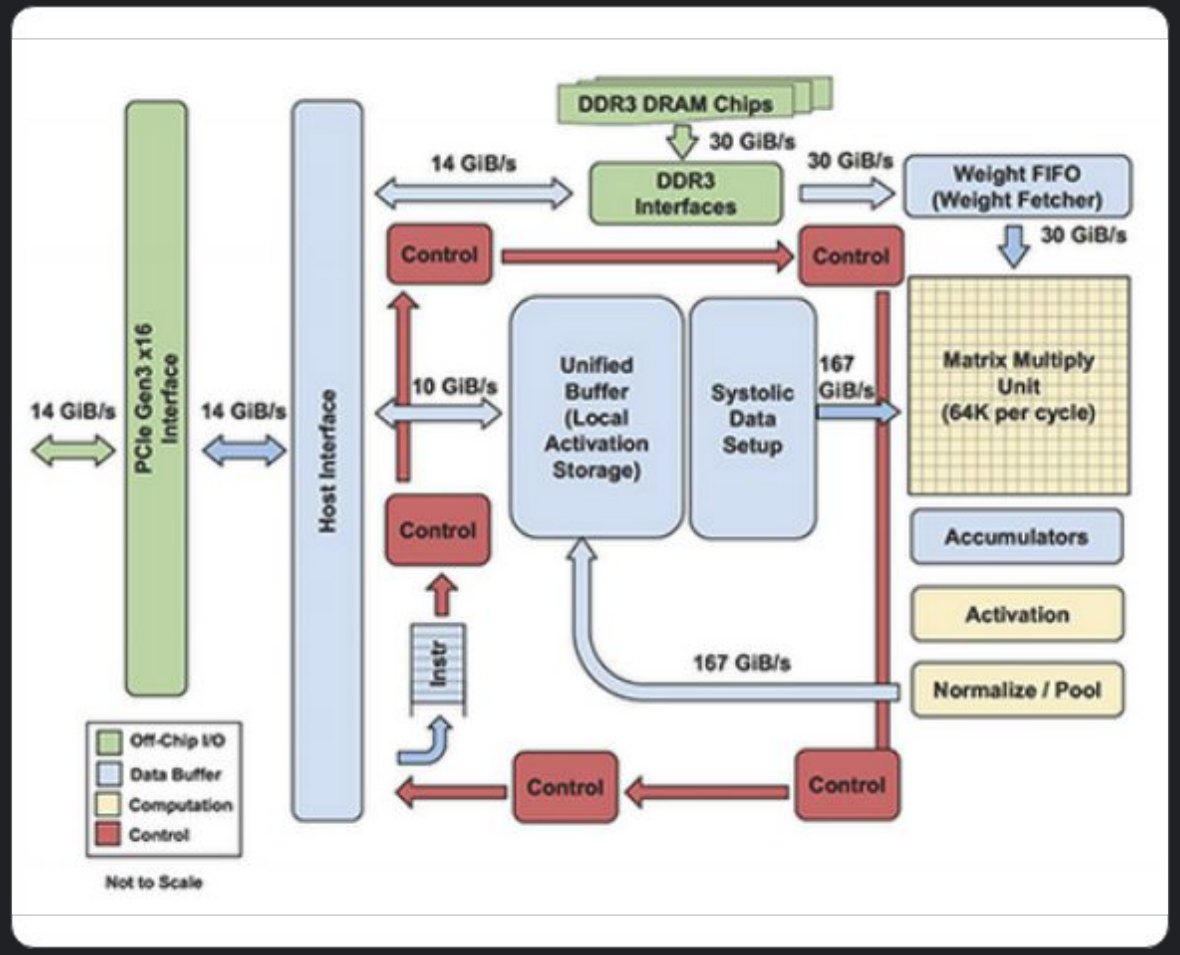
Google Cloud TPUs

# Google Cloud TPUs

While GPUs are great multi-purpose tools, **Tensor Processing Units (TPUs)** are custom-built by Google specifically for AI workloads.
TPU v6 pods offer exascale performance designed to train massive foundation models efficiently.



# Why TPUs for Bio-AI?



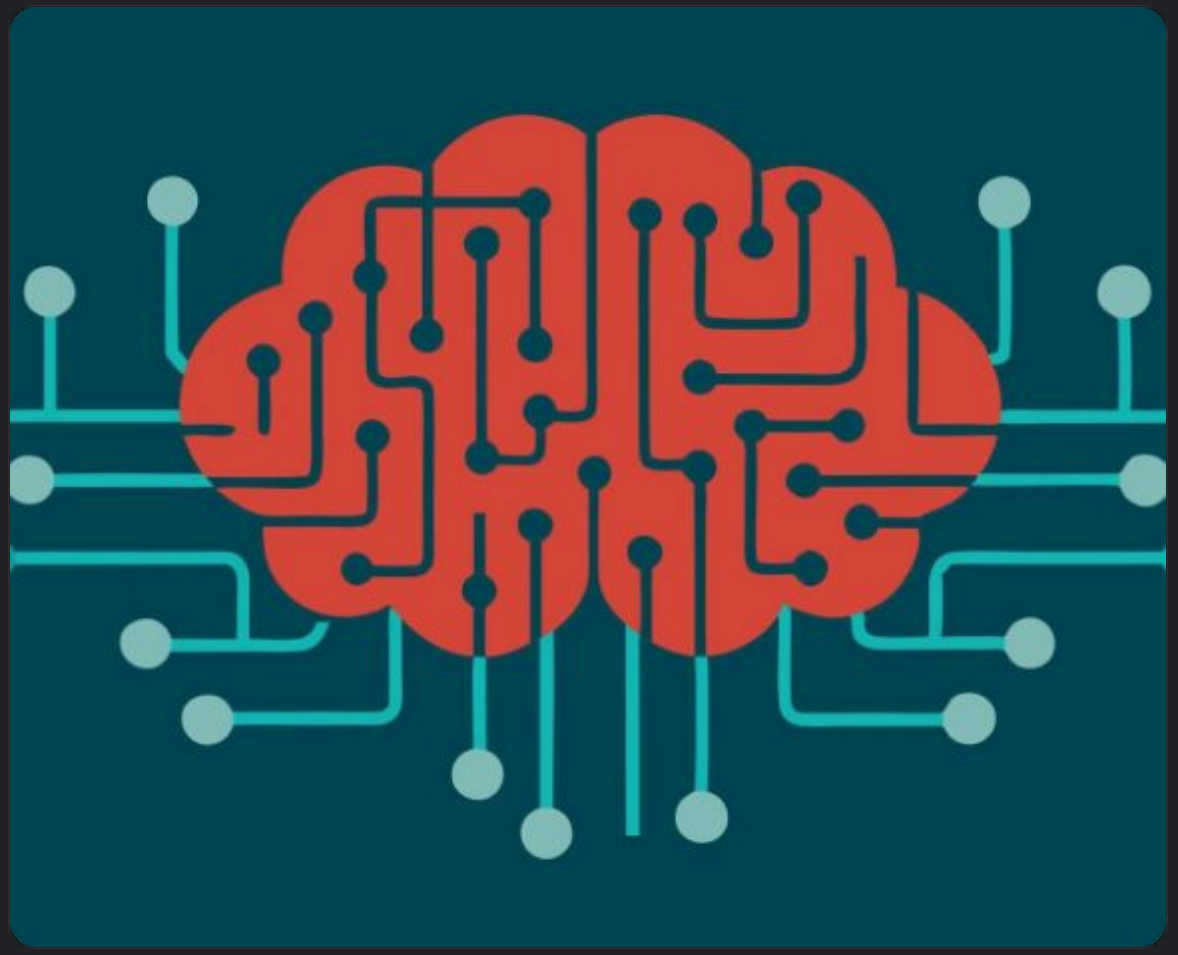

The Systolic Array

- Unlike GPUs that process many diverse tasks, TPUs use a "systolic array" design highly optimized for massive matrix multiplications (the core of all AI).
- Data flows through the chip like a heartbeat, maximizing throughput for genomic transformer training.

# The Future: AI Writing HPC Software

Optimizing these kernels (like FlashAttention) is incredibly hard.

The future is using AI itself to write highly-optimized CUDA or Triton kernels, automatically tailoring software to new biological model architectures.



# Key Takeaways

Software is Key
HPC power is useless without compilers (XLA) and fusion to reduce overhead.

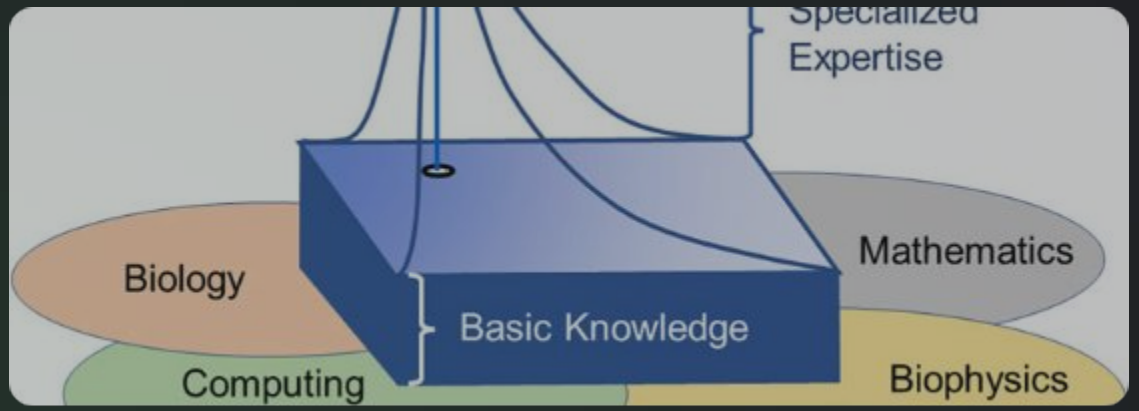
Democratization
Tools like QLoRA and Keras 3 allow any lab to use state-of-the-art AI.

Specialization
Hardware like TPUs are purpose-built for the matrix math that defines modern Bio-AI.

Algorithm Innovation
FlashAttention and new data types (NF4) are just as important as new chips.

# Thank You

## Questions?

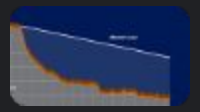


# Image Sources

http://mw.concord.org/modeler/_assets/img/three-pillars.png

Source: mw.concord.org

---



https://www.genome.gov/sites/default/files/inline-images/2022_Sequencing_cost_per_Mb.jpg

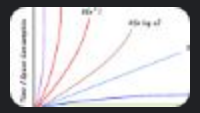
Source: www.genome.gov

---

https://docs.graphcore.ai/projects/ipu-programmers-guide/en/3.3.0/_images/software-stack.png

Source: docs.graphcore.ai

---



https://miro.medium.com/0*Re796fmjO90us8op.png

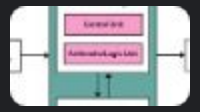
Source: medium.com

---

https://cdn.prod.website-files.com/687b2d16145b3601a227c560/68d2b2eea959bbc1c75ffe33_68ce86e4319e6069aed4aa0b_68b16ffcbcfca56d339ce067_1756421718769-ecshj.jpeg

Source: airbyte.com

---



https://upload.wikimedia.org/wikipedia/commons/e/e5/Von_Neumann_Architecture.svg

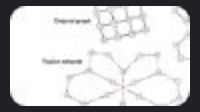
Source: wikipedia.org

# Image Sources

https://developer-blogs.nvidia.com/wp-content/uploads/2023/08/gromacs-kernel.png

Source: developer.nvidia.com

---



https://quantum-journal.org/wp-content/uploads/2023/12/main_figure-1.png

Source: quantum-journal.org

---

https://openxla.org/xla/images/gpu_pipeline.png

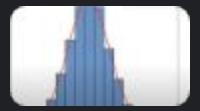
Source: openxla.org

---

https://developer-blogs.nvidia.com/wp-content/uploads/2021/07/qat-training-precision.png

Source: developer.nvidia.com

---



https://miro.medium.com/1*U1SbZhYcEGMBAOQ2ilJ64A.png

Source: athekunal.medium.com

---

https://miro.medium.com/v2/resize:fit:1400/1*rOW5plKBuMlGgpD0SO8nZA.png

Source: abvijaykumar.medium.com

# Image Sources

https://machinelearningmastery.com/wp-content/uploads/2018/12/Example-of-Train-and-Validation-Learning-Curves-Showing-a-Training-Dataset-the-May-be-too-Small-Relative-to-the-Validation-Dataset.png

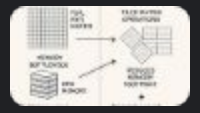
Source: machinelearningmastery.com

---

https://miro.medium.com/1*lWWsrvUtotUm8TNiy75w7w.png

Source: medium.com

---



https://miro.medium.com/v2/resize:fit:1400/1*g7bBOsaHM3lM61R8hvkk3g.png

Source: medium.com

---

https://miro.medium.com/v2/resize:fit:2000/1*i4tDdwgvGtXuTlyJpFUn8A.png

Source: gordicaleksa.medium.com

---


https://storage.googleapis.com/lightning-avatars/litpages/01hw55j5mtcpyw5a4yxwsggs0c/1c1963ca-2909-4dc2-b0d6-4a33b18f6255.jpg

Source: lightning.ai

---

https://media.jax.org/m/c6c582dbee9353d0/webimage-genetic-engineer-final-final-final-2024.png

Source: www.jax.org

# Image Sources

https://storage.googleapis.com/gweb-cloudblog-publish/images/TPU_v5L_Pod_-_Front_View_-_Web.max-2600x2600.jpg

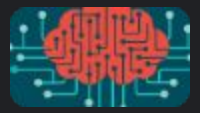
Source: cloud.google.com

---

https://storage.googleapis.com/gweb-cloudblog-publish/images/tpu-15dly1.max-500x500.PNG

Source: cloud.google.com

---



https://static.vecteezy.com/system/resources/previews/060/289/390/non_2x/brain-and-circuit-an-abstract-representation-of-a-brain-composed-of-circuitry-lines-signifying-the-intelligence-of-ai-in-processing-and-writing-code-vector.jpg

Source: www.vecteezy.com

---

https://www.frontiersin.org/files/Articles/1250228/fsysb-03-1250228-HTML/image_m/fsysb-03-1250228-g003.jpg

Source: www.frontiersin.org