Hardware Acceleration

CS6501: AI Systems

Fall 2026

Lecture 4

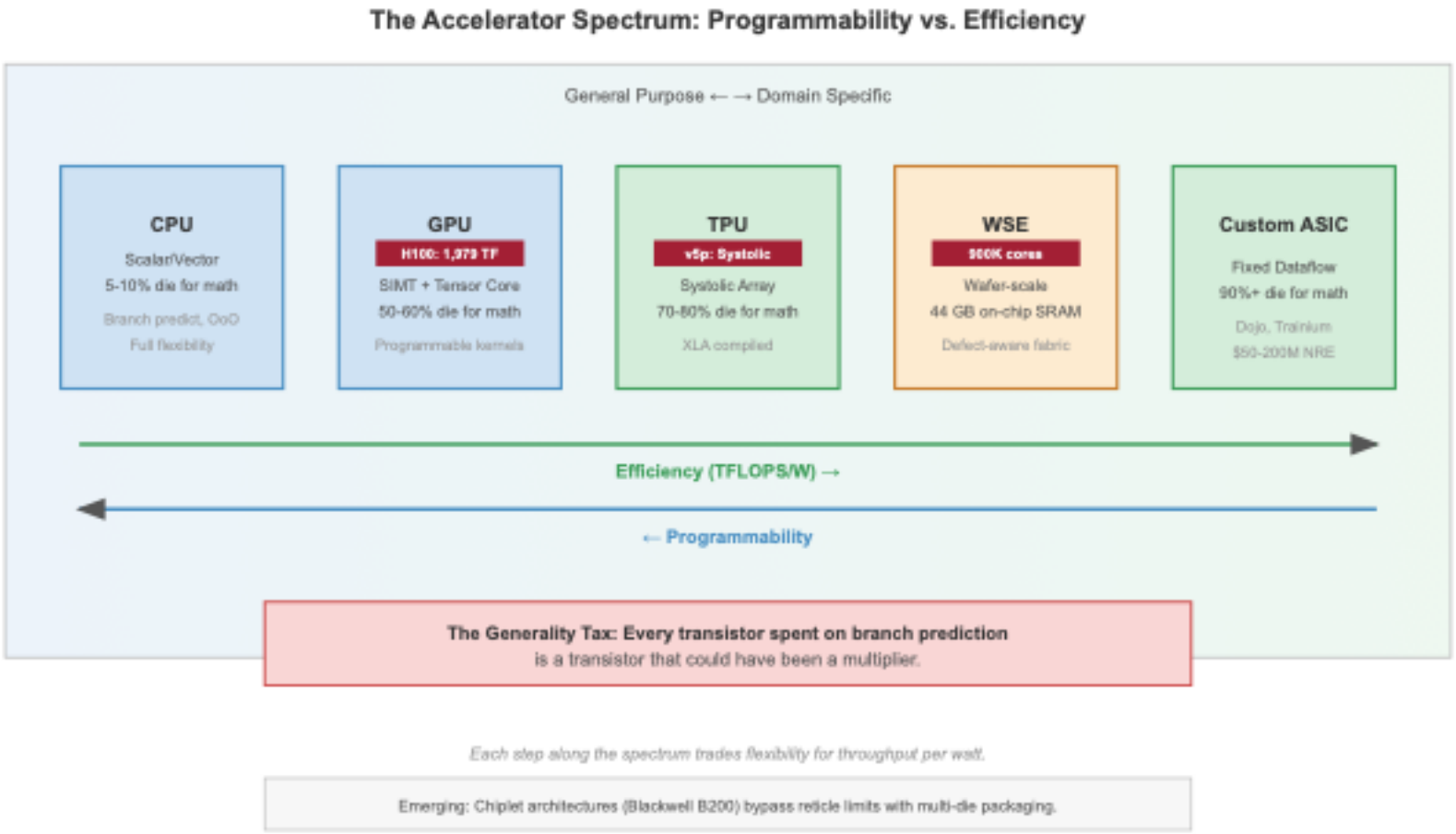
Yue Cheng

Material taken/derived from:

- Harvard ML Systems by Vijay Janapa Reddi
- OpenAI Jalapeño Hot Chips 2026 talk

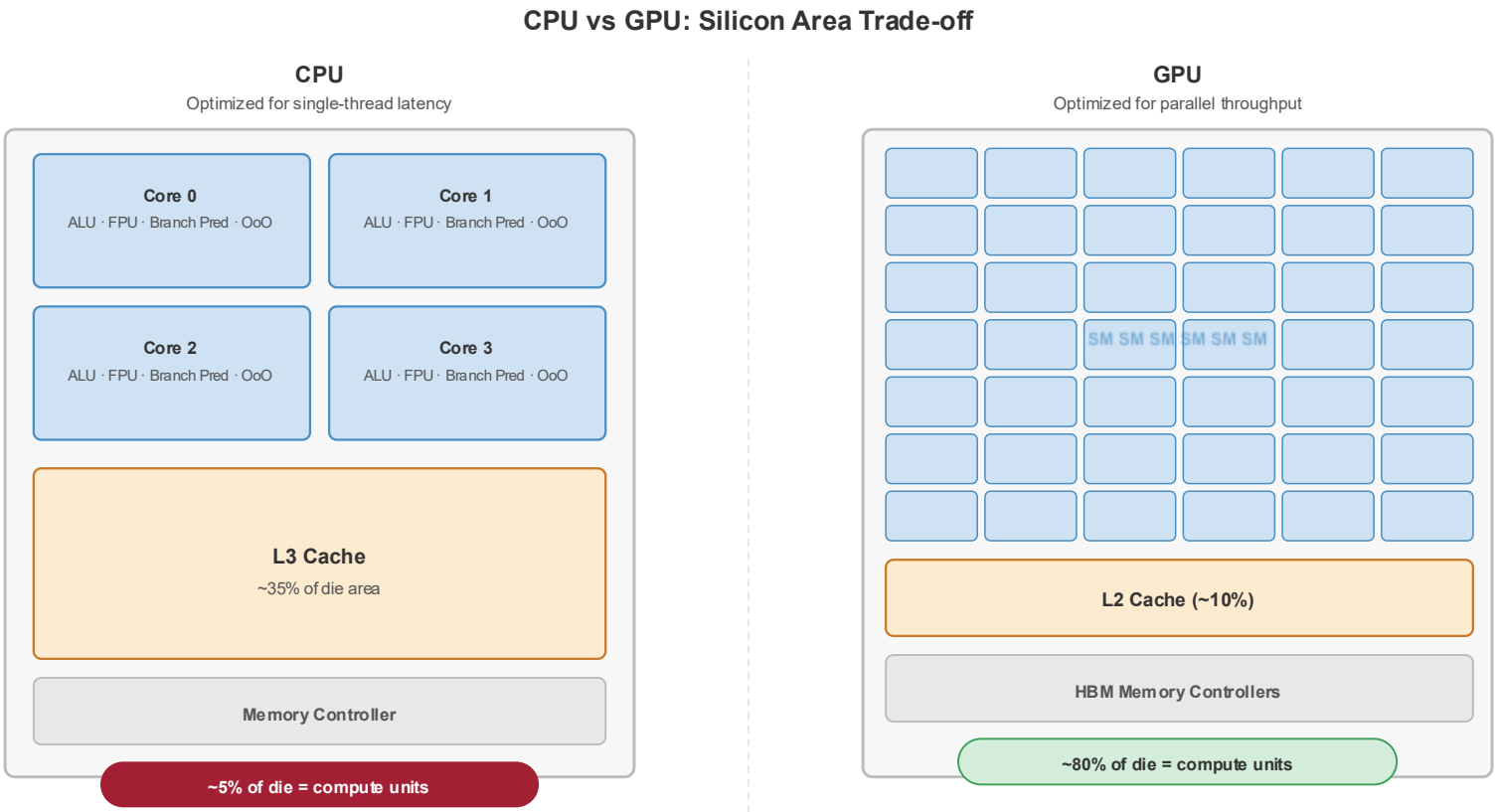
Why specialize?

The accelerator spectrum



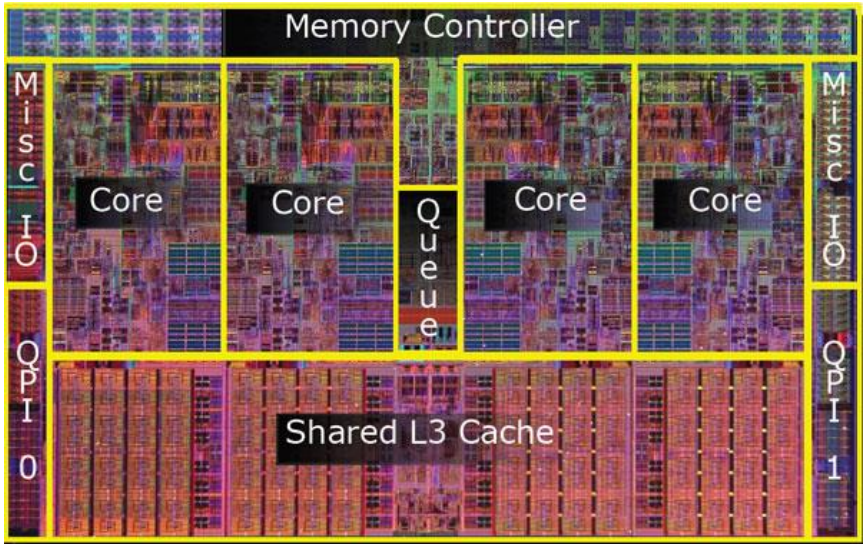
The generality tax: Every transistor spent on a branch prediction is a transistor that could have been a multiplier.

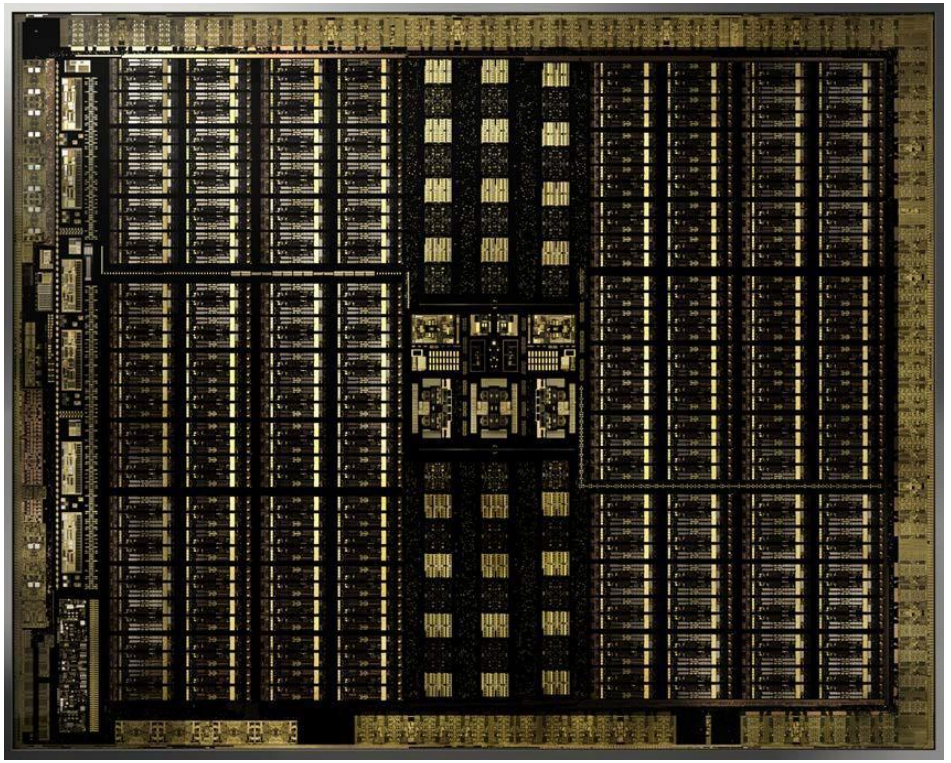
CPU vs. GPU: The silicon area tradeoff



GPUs trade single-thread latency for massive parallel throughput by dedicating die area to compute rather than caches.

CPU vs. GPU: The silicon area tradeoff



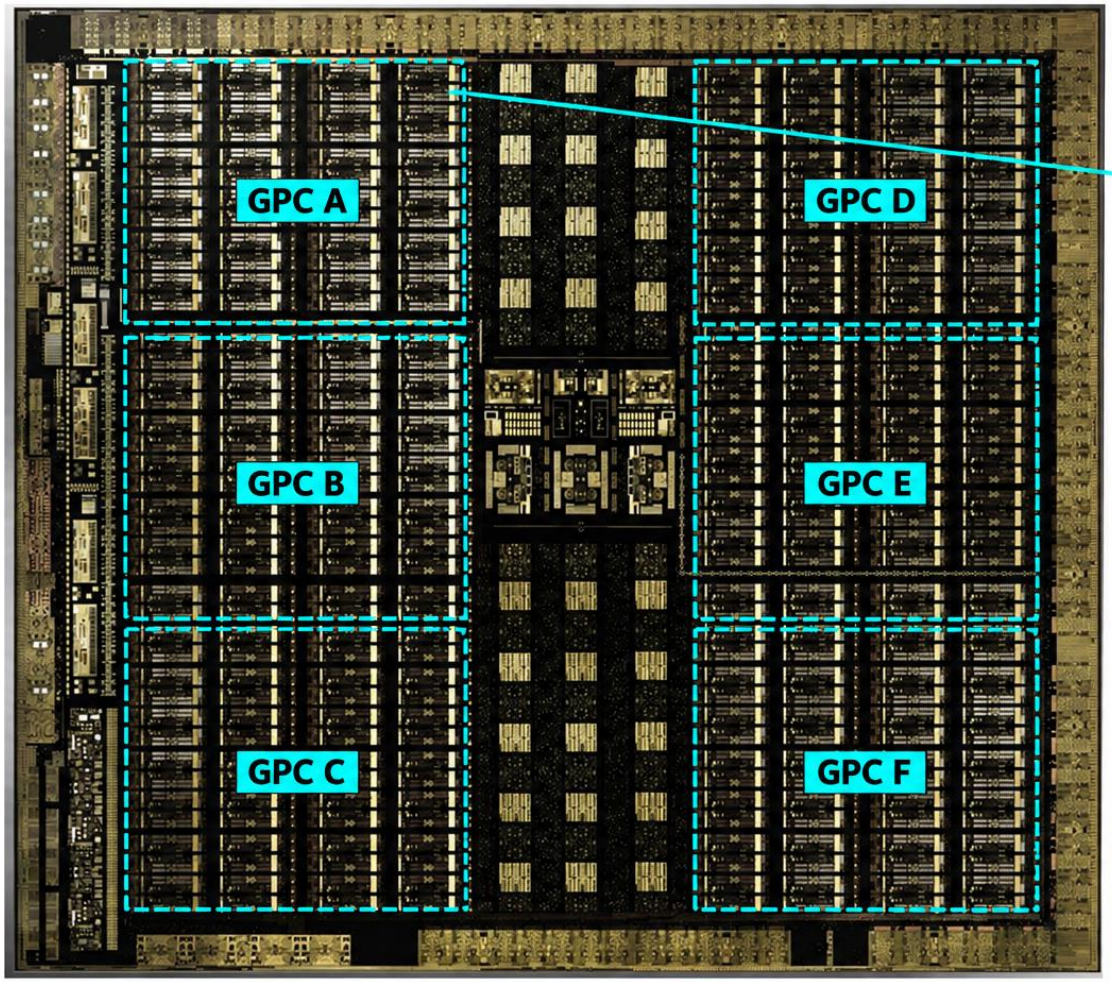


CPU: ~5% compute
Intel i7 core die*

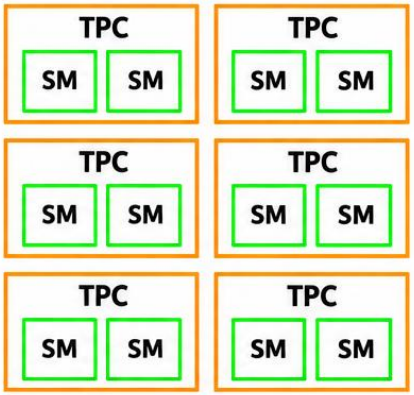
GPU: ~80% compute
NVIDIA Turing*

*: https://pcper.com/2008/08/inside-the-nehalem-intels-new-core-i7-microarchitecture/
*: https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/

NVIDIA TU102: GPC, TPC, and SM hierarchy



Inside one GPC (schematic)



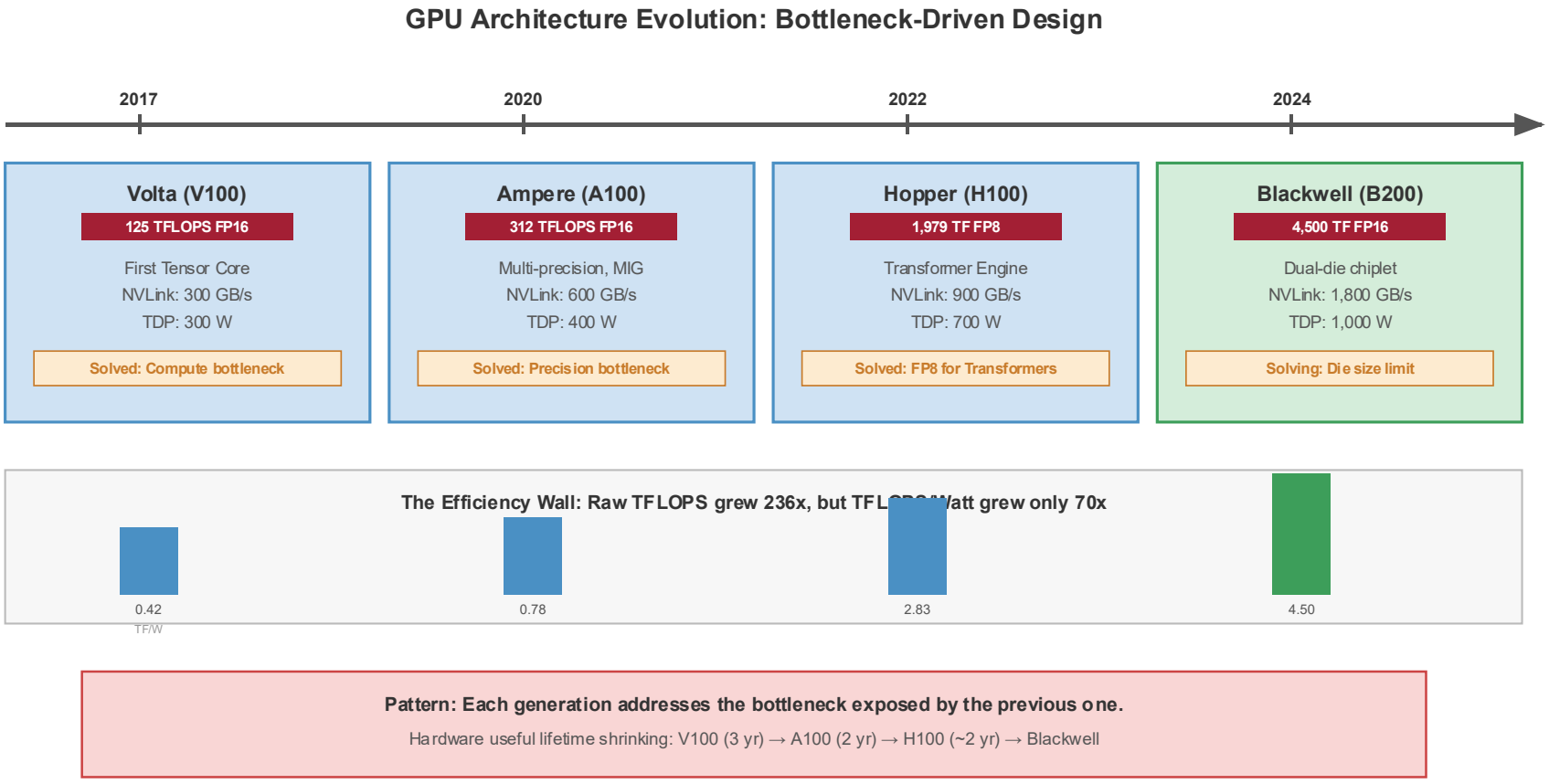
6 GPCs × 6 TPCs × 2 SMs = 72 SMs

GPC	= Graphics Processing Cluster
TPC	= Texture Processing Cluster
SM	= Streaming Multiprocessor

GPC outlines are approximate. TPC/SM inset shows logical hierarchy, not verified die boundaries.

Source: NVIDIA Turing Architecture In-Depth

GPU architecture evolution



GPU architects profile production ML workloads to identify the next bottleneck. The workload drives the silicon.

GPU specs: Four generations

Arch.	Year	Peak TF	TDP	NVLink	TF/W
Volta (V100)	2017	125	300 W	300 GB/s	0.42
Ampere (A100)	2020	312	400 W	600 GB/s	0.78
Hopper (H100)	2022	1,979	700 W	900 GB/s	2.83
Blackwell (B200)	2024	4,500	1,000 W	1,800 GB/s	4.50

The efficiency wall: Raw TFLOPS grew 236x (P100 → B200), but TFLOPS/Watt grew only 70x. The gap is what cooling and power infrastructure must absorb.

Amdahl's law: The accelerator ceiling

$$\text{Speedup} = \frac{1}{(1-p) + \frac{p}{S}}$$

p	Parallel fraction (matrix math)
S	Accelerator speedup (100—1000×)
$1-p$	Serial fraction (data loading, Python, etc.)

Workload	p	Max Speedup
ResNet-50	0.95	20×
GPT-2 (gen.)	0.80	5×
DLRM (embed)	0.60	2.5×
MobileNet	0.70	3.3×

Key: ResNet scales better – more time in parallelizable matrix math.

Pitfall: If data loading is 10% ($p = 0.9$), even $S = \infty$ yields only 10× total speedup.

Predict: Where is the bottleneck?

Q: A transformer model runs on an H100 GPU. `MatMul` layers hit 280 TFLOPS. `Softmax` layers hit 6 TFLOPS.

Why does the same GPU give such different throughput for operations in the same model?

The central surprise of modern computing

The memory wall:

- In the time to fetch **one value** from DRAM, an accelerator performs **1,000× operations**
- DRAM access costs **20,000×** more energy than an INT8 add
- This is not engineering – it's **physics**

The question is never "which chip is fastest?" but **"which memory system best matches my access patterns?"**

Compute primitives

The MAC: NNs' dominant operation

Three compute primitives dominate all accelerators:

Primitive	Example	Hardware
Vector ops	ReLU, bias add	SIMD units
Matrix ops	Dense, Conv2D	Tensor Cores
Special fn	Softmax, GELU	SFUs

Key: MACs are 95%+ of execution time.

Dense layer (256→512)

```
for n in batch:
  for m in output:
    sum = bias[m]
    for k in input:
      sum += x[n,k]*w[k,m]
```

$32 \times 512 \times 256 = 4.2\mathbf{M\ MACs}$

Tensor Cores: GPU's matrix engine

NVIDIA Tensor Core (since Volta, 2017):

- V100: **640 Tensor Cores** → 125 TF (FP16)
  - Each core: $4 \times 4 \times 4$ matrix MAC per clock
- A100: **432 Tensor Cores** → 312 TF (FP16)
  - Each core: 256 matrix MAC per clock
- H100: **528 Tensor Cores** → 989 TF (FP16)
  - Each core: 512 matrix MAC per clock

Peak utilization requires: FP16/BF16 inputs, dimensions aligned to 8/16, sufficient batch size.

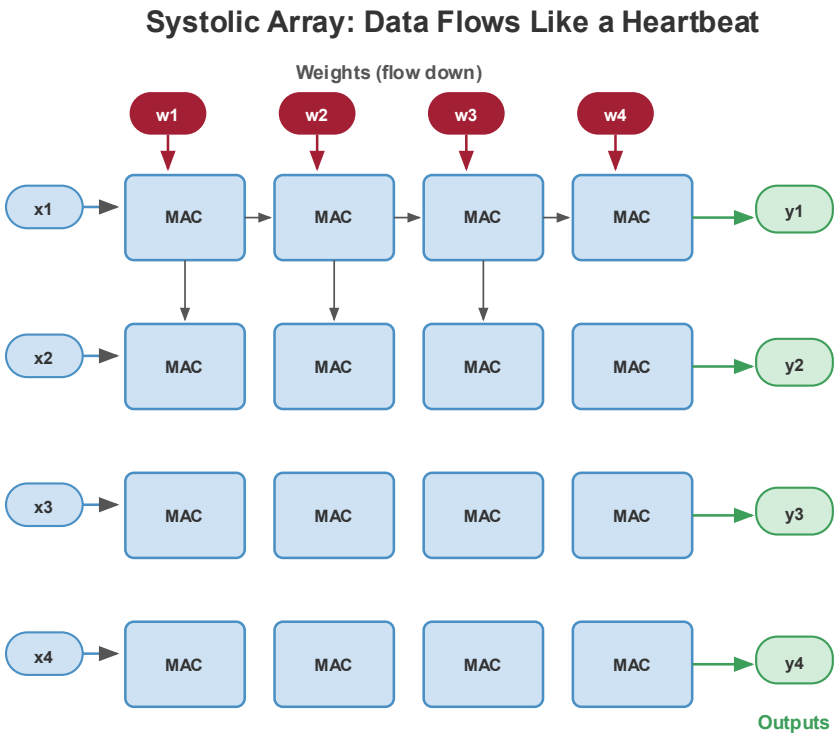
1,000× growth in one decade (K20X → H100) – specialization, not clock speed.

Google TPUs

- In 2013, Google engineers projected if every user spoke to their Android phone for just 3 min per day using voice search, Google would need to **double its datacenter compute capacity**
- Over **90%** of inference cycles were spent on matrix multiplications in NNs
- GPUs are powerful but **cost prohibitive**
- General-purpose SIMT processors carried the **generality tax**

TPU printed circuit board. It can be inserted into the slot for a SATA disk in a server.

Systolic arrays: Data reuse at scale



TPUv1: 256 x 256 Systolic Array

65,536 MACs operating simultaneously

Each weight loaded once, used 256 times

Data reuse is the key to efficiency

Why "Systolic"?

Named after the heart's rhythmic pumping.
Data ripples through the array like
blood through the circulatory system.

Without systolic reuse:

Each weight fetched from DRAM every cycle
= 256x more memory bandwidth required

Data reuse: TPUv1's 256×256 array = 65,536 MACs. Each weight loaded once, used 256 times

The memory wall

The memory wall
(Wulf & McKee, UVA, 1994)

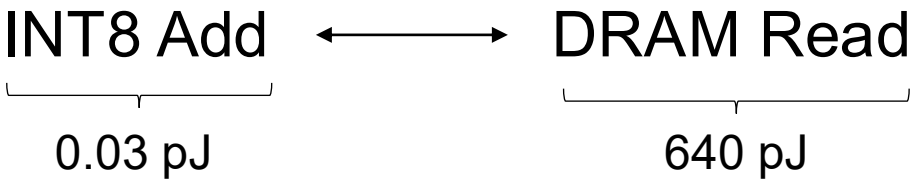
Processor speed improves much faster than DRAM speed.

Performance becomes memory-bounded.

* Hitting the memory wall: Implications of the obvious. Wm Wulf and Sally McKee. CS UVA tech report (1994) https://doi.org/10.18130/V3FZ1C

The cost of data movement
(Horowitz, 2014)

Moving data costs **20,000**× more energy than computing it.



This is physics, not engineering.

* Compute's energy problem. Mark Horowitz. ISSCC 2004.

The Horowitz numbers: Energy per op

Horowitz, ISSCC 2014 (45 nm reference):

Operation	Width	Energy (pJ)	Ratio
INT8 Add	8-bit	0.03	1×
INT8 Multiply	8-bit	0.2	7×
FP16 Multiply	16-bit	1.1	37×
FP32 Multiply	32-bit	3.7	123×
SRAM Read	32-bit	5	167×
DRAM Read	64-bit	**640**	**20,000×**

Dense layer energy split (INT8):

1M MACs × 3.7pJ = 3.7μJ

1M DRAM reads × 640 pJ / 8 = 80 μJ

Data movement: 95.6%

Compute: 4.4%

Implication: Reducing data movement by 2× saves more energy than making ALUs 20× faster.

Token latency: The physics

Scenario: One token of Llama-3 70B on H100

Component	Calculation	Time	Share
Compute	$2 \times 70B/989$ TF	**0.14 ms**	$<1\%$
Memory	140 GB/3.35 TB/s	**41.8 ms**	$>$**99%**
		$T_{token} \approx$	**41.9 ms**

The processor spends more than 99% of this idealized token step waiting for data. Even a hypothetical chip with infinite compute would generate tokens only negligibly faster. **Memory bandwidth is the binding constraint.**

Question: Memory-bound or compute-bound?

A team trains a 175B model at batch size 2,048. Another team serves the same model at batch size 1.

Which workload benefits more from buying a faster GPU (more TFLOPS)?

Roofline

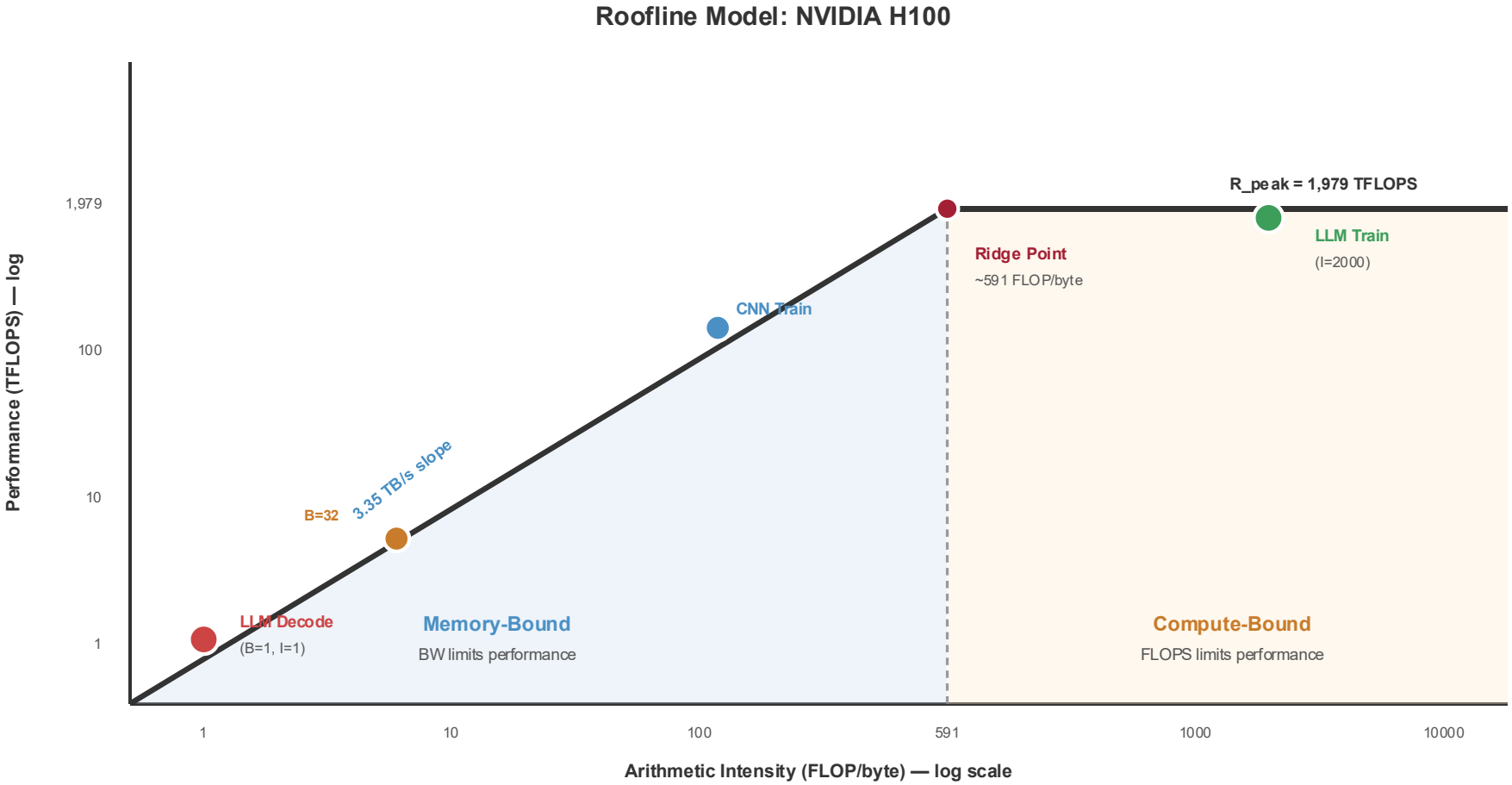
The Roofline model

$$\text{Achievable FLOPS} = \min(R_{peak}, \text{BW} \times I)$$

where I = FLOP/byte (arithmetic intensity)

* Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures. Samuel Webb Williams, Andrew Waterman, David A. Patterson. EECS UC Berkeley tech report (2008) https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/Archive/EECS-2008-134.pdf

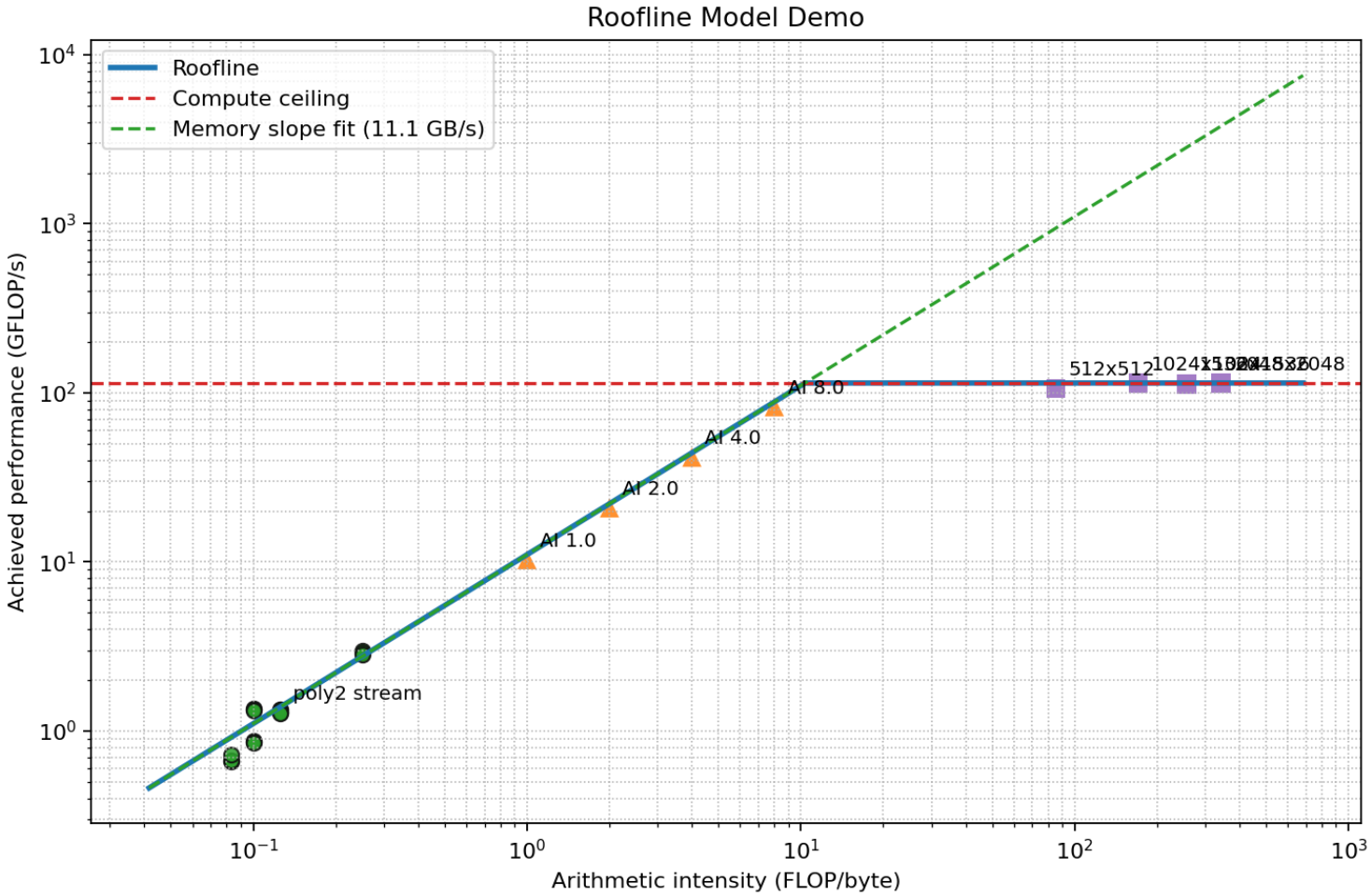
Roofline: H100 landscape



LLM decode achieves <1% of peak FLOPS. The same hardware that excels at training appears woefully underutilized during inference.

Diagnostic power: The Roofline tells you which resource to optimize before you spend money.

Roofline: EC2 t3.large



Roofline: Workload placement

Workload	/ (FLOP/B)	% Peak	Regime
LLM decode (B=1)	∼1	<1%	**Memory-bound**
LLM decode (B=32)	∼32	∼5%	Memory-bound
CNN training	50–200	30–60%	Near ridge
LLM prefill	100–500	50–80%	Transitioning
LLM training (175B)	∼2000	30–50%	**Compute-bound**

Below ridge: Buy bandwidth
Faster GPU is wasted money

Above ridge: Buy compute
Faster HBM is wasted money

Roofline: Workload placement

Workload	/ (FLOP/B)	% Peak	Regime
LLM decode (B=1)	∼1	<1%	**Memory-bound**
LLM decode (B=32)	∼32	∼5%	Memory-bound
CNN training	50–200	30–60%	Near ridge
LLM prefill	100–500	50–80%	Transitioning
LLM training (175B)	∼2000	30–50%	**Compute-bound**

Below ridge: Buy bandwidth
Faster GPU is wasted money

Above ridge: Buy compute
Faster HBM is wasted money

Improve data locality or effective bandwidth

Improve compute throughput or utilization

Your turn: Roofline diagnosis

Diagnose the bottleneck

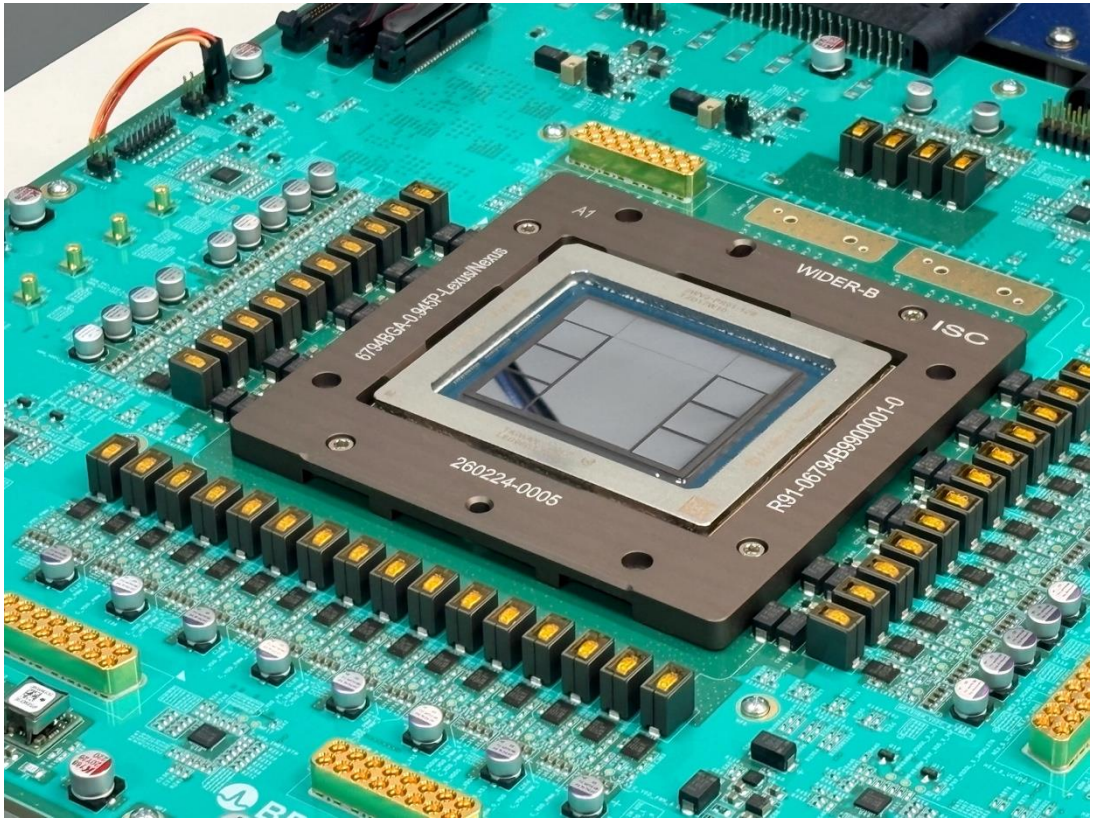
A 7B model (14 GB FP16 weights) served at batch size 1 on an H100

- R_{peak} = 1,979 TFLOPS
- BW = 3.35 TB/s
- Ridge point = ~591 FLOP/byte

Calculate the arithmetic intensity and determine the regime

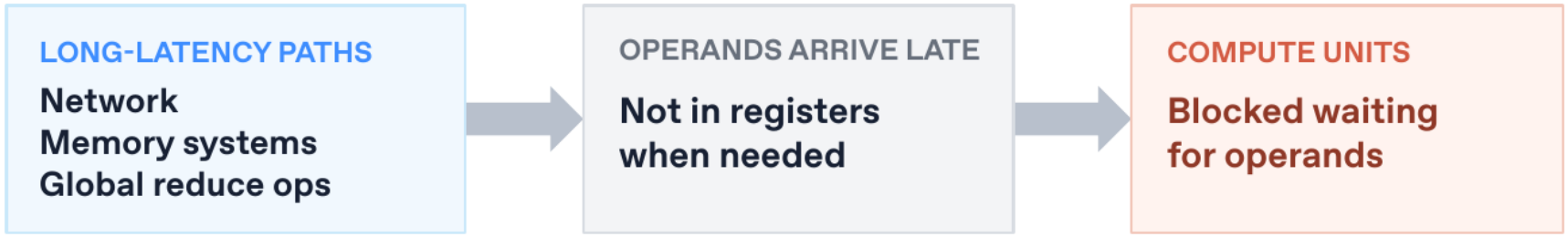
Jalapeño: LLM inference

OpenAI's custom accelerator for LLM inference



* Jalapeño's first results. OpenAI. 2026. https://openai.com/index/jalapeno-first-results/

Architecture can dominate observed latency



What causes long latencies and late operands?

Unified memory subsystems tend to highly contended paths
Independent cores out of sync make global memory fences expensive
Centralized resources mediating network access

→ NOT hardware speed of light

Utilization falls when compute/memory blocked waiting on data in transit

OpenAI

* Hot Chips 2026. Jalapeño slide #24.

Metrics that matter: latency and energy

01 · USER EXPERIENCE

Request latency

Time to complete an inference request

TTLT · time to last token

End-to-end latency; TBT is another useful proxy

02 · INFERENCE EFFICIENCY

Energy per request

Energy needed to process an inference request

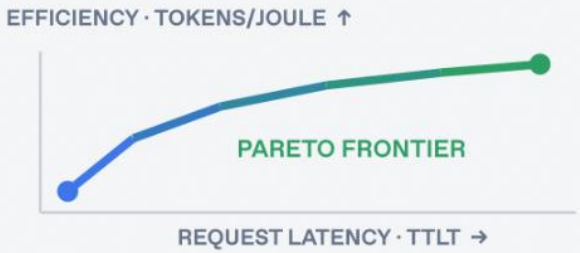
Energy / token · tokens / joule

Also measured as tokens/s/kW

Single user latency ↔ energy per token

Lower latency usually costs more energy per token.
Energy is a proxy for cost

Compare systems across the full Pareto frontier.



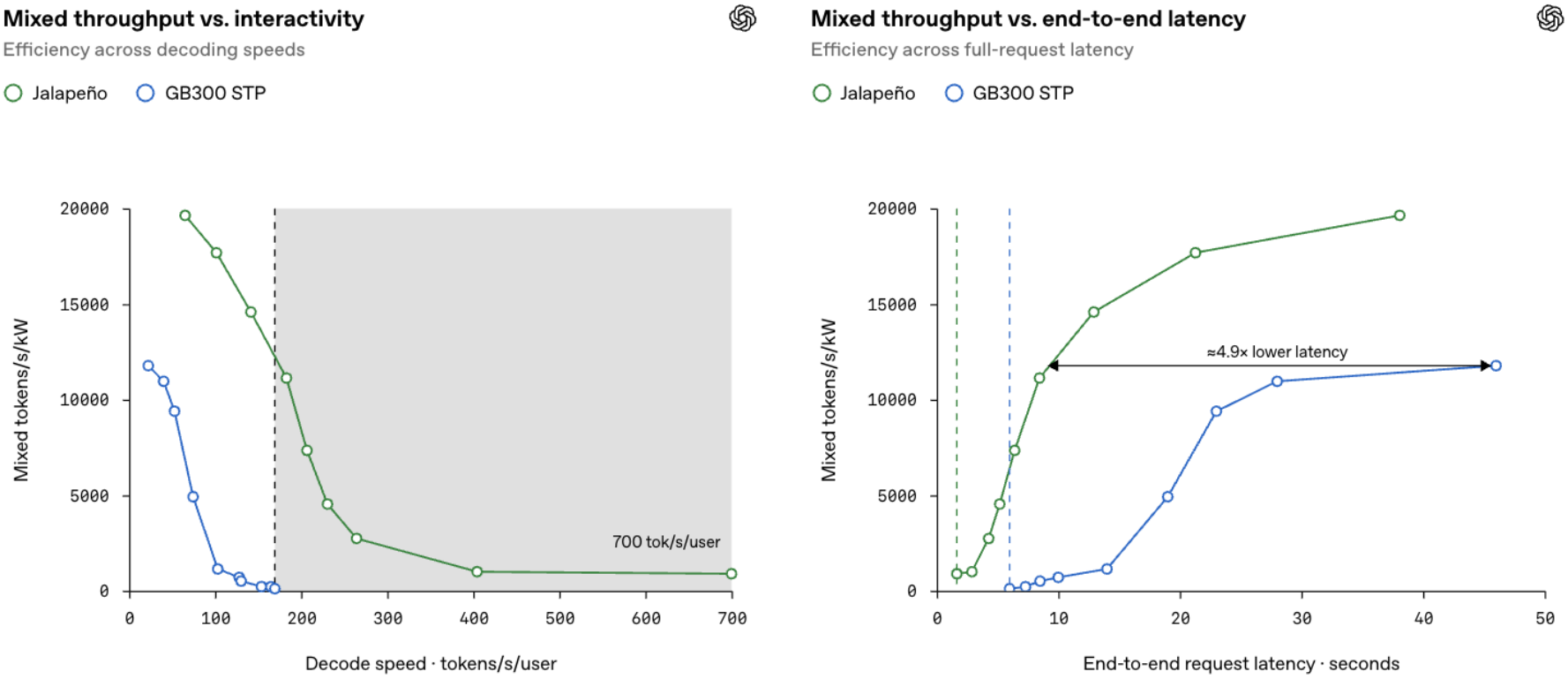
NON GOALS ~~# chips used~~ ~~throughput per chip~~ ~~TTFT~~

Compare efficiency @ matched user experience

OpenAI

* Hot Chips 2026. Jalapeño.

Jalapeño is pareto frontier at DeepSeek R1 670B



THROUGHPUT-per-kW FRONTIER

InferenceX · DeepSeek R1 MXFP4 · nominal 8k/1k · STP · package TDP: Jalapeño 700 W; GB300 1,400 W

* Jalapeño's first results. OpenAI. 2026. https://openai.com/index/jalapeno-first-results/

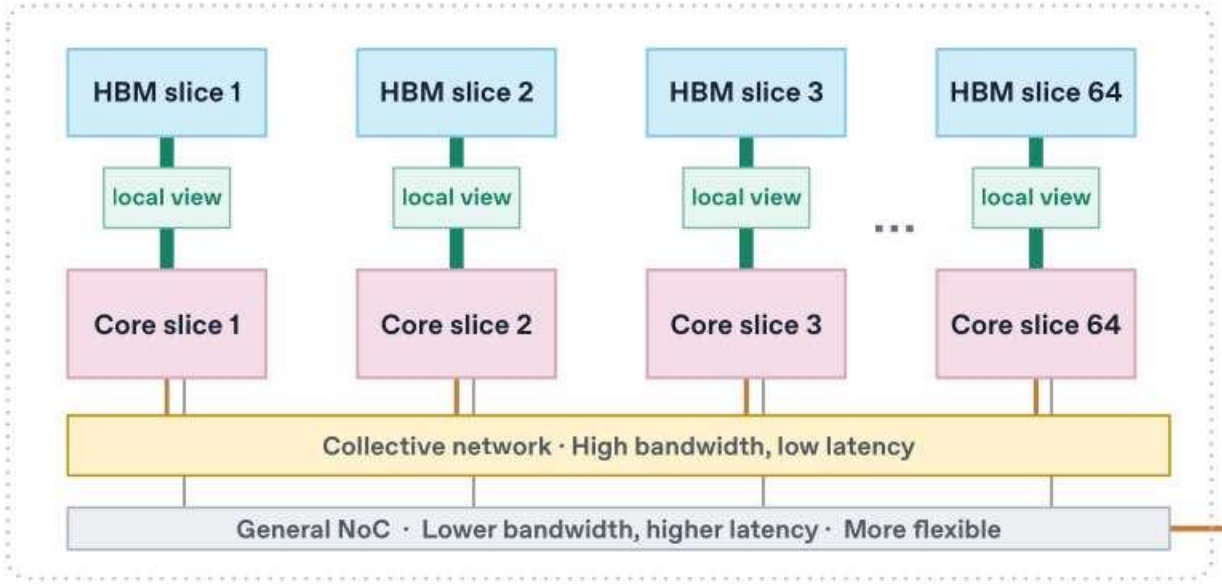
Jalapeño: LLM inference

OpenAI's custom accelerator for LLM inference

Prefill	**Decode (small batch)**
Many input tokens at once	Once new token per sequence
More weight reuse	Less weight reuse
Higher arithmetic intensity	Lower arithmetic intensity

One model, different resource need

Co-design the architecture to address the bottleneck



FASTEST LOCAL PATH

Pair each core slice with an HBM slice to expose a low-latency, high-bandwidth local view.

SPECIALIZED CROSS-CORE

Collective network optimizes bandwidth and latency of common comms patterns.

PRESERVE FLEXIBILITY

Shared NoC for general comms and access to scale-up network.

Keep common operands local; use the shared path only when necessary

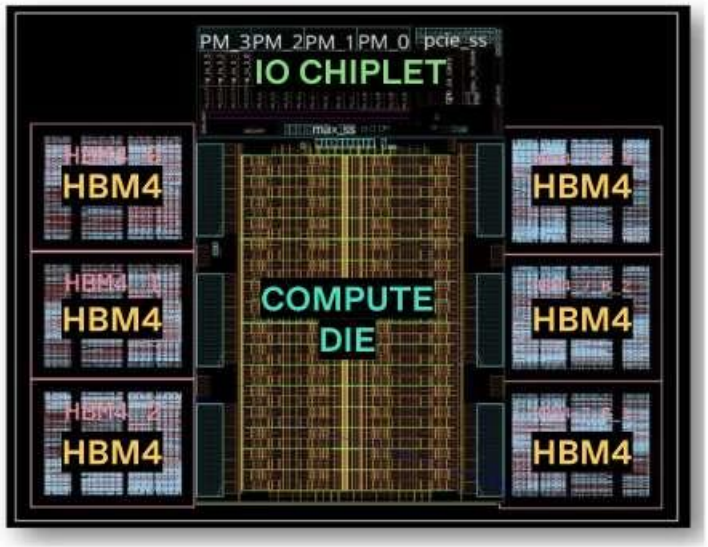
Explicit placement and distributed control prevent data movement/synchronization from dominating execution

OpenAI

* Hot Chips 2026. Jalapeño slide #25.

Specifications

Jalapeño ASIC	
Matrix compute	mxfp8 x mxfp8 = 3.4 PFLOP/s mxfp8 x mxfp4 = 6.7 PFLOP/s mxfp4 x mxfp4 = 13.4 PFLOP/s
Memory system	15.4 TB/s, 216 GiB
Scale-up network	Local = 128 ASICs @ 600 GB/s Global = 2048 ASICs @ 200 GB/s
Power	700 W
Jalapeño system (2048 ASICs)	
Matrix compute	mxfp4 x mxfp4 = 27 EFLOP/s
Memory system	32 PB/s, 432 TiB



Package floorplan

Most important spec: perf/W on end-to-end workloads

OpenAI

32

* Hot Chips 2026. Jalapeño slide #32.