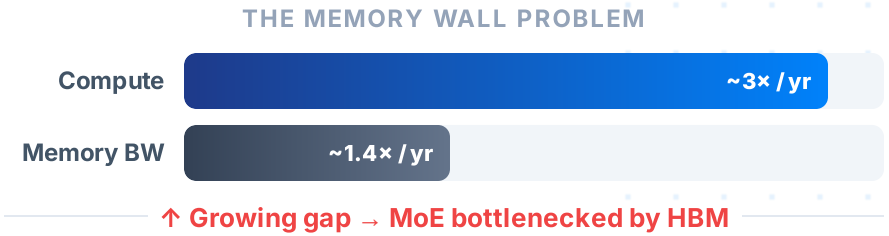
MLSYS 2026

MoEBlaze: Breaking the Memory Wall for Efficient MoE Training on Modern GPUs

MoE Training | Memory Efficiency | Kernel Fusion | GPU Systems

AUTHORS

Jiyuan Zhang · Yining Liu · Siqi Yan
Lisen Deng · Jennifer Cao · Shuqi Yang
Min Ni · Bi Xue · Shen Li



4–6× Expert compute speedup

2× Dispatch speedup

2M+ GPU hours saved

Meta

The Memory Wall is Widening

The 20-Year Gap

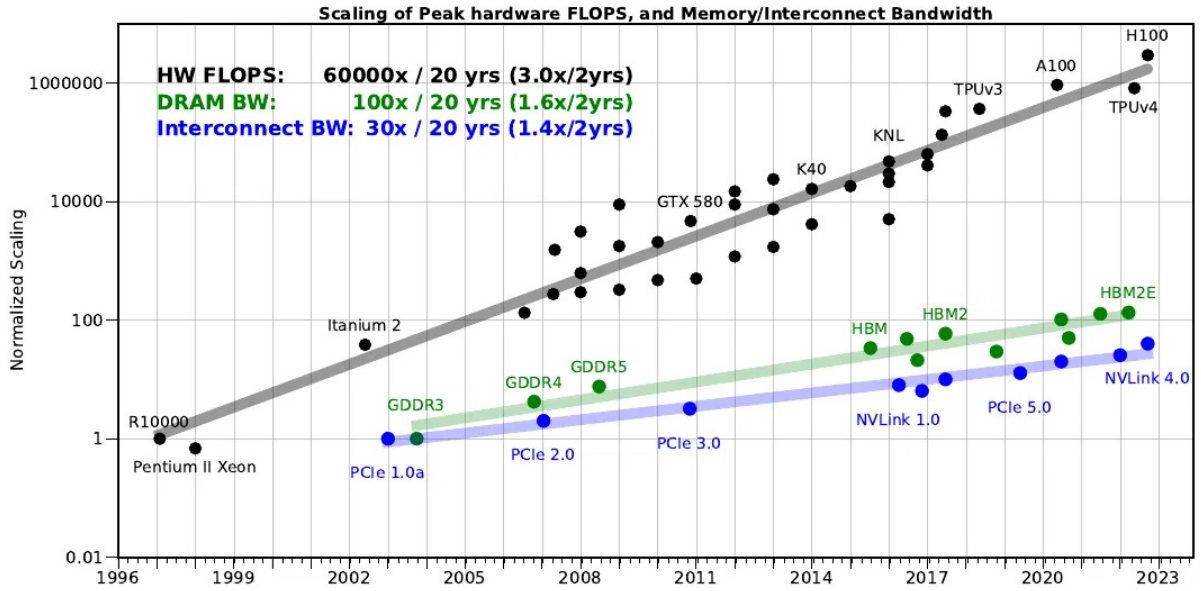
60,000x

Hardware Compute Growth

vs.

100x

DRAM Bandwidth Growth



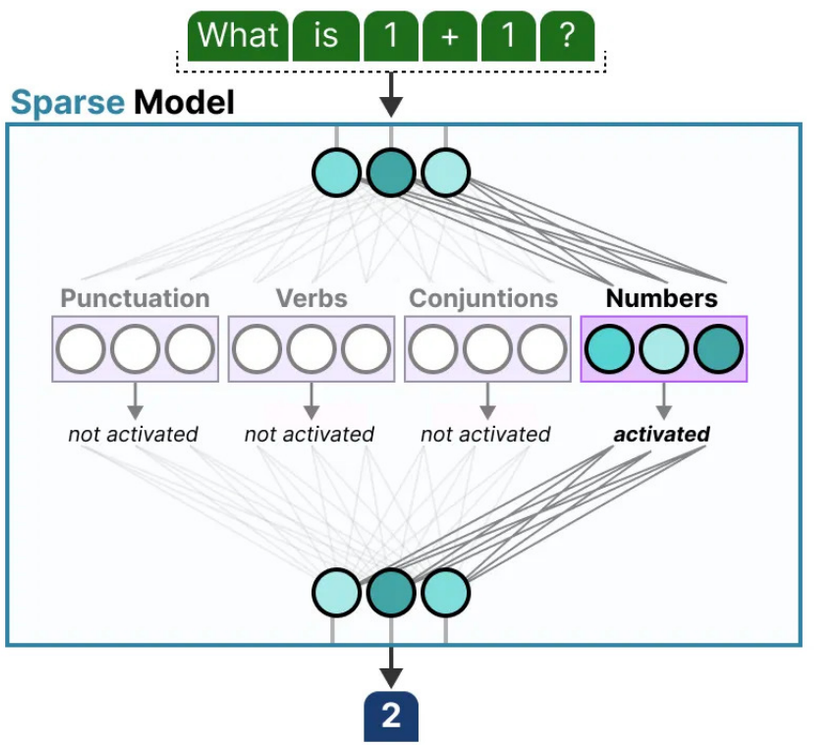
Source: AI and Memory Wall (Gholami et al., IEEE Micro 2024)

Result: Data movement, rather than compute, is the primary bottleneck.

MoE Architecture & The Cost of Sparsity

Parameter Efficiency via Sparse Activation

Mixture-of-Experts (MoE) scales model capacity without proportional compute increases. A router network dynamically selects only a small subset of experts to process each token.



Sparsity Comes at a Cost

While highly compute-efficient, the MoE architecture severely amplifies the **Memory Wall** bottleneck.

This bottleneck stems from two primary sources:

- **Massive Intermediate Storage:** Dynamic routing and complex activations require materializing huge intermediate tensors.
- **Low Arithmetic Intensity:** The ratio of compute operations to memory bytes accessed is extremely low, leading to poor hardware utilization.

> **These factors combine to starve compute units, leaving modern GPUs severely underutilized.**

Source 1: Memory Capacity Bottleneck

A Single MoE Layer's Footprint

In conventional implementations, activation memory is driven by two massive components:

- **Token Routing Buffer:** Auxiliary per-expert buffers needed to compact and store dispatched tokens.
- **FFN Intermediate Activations:** Storage amplified by modern non-linearities (like SwiGLU) which require materializing intermediate tensors.

DeepSeek-Scale Example

Assuming $L \approx 2$ *million* tokens, $K = 4$ active experts, $d = 6144$, and $h = 24576$ (bfloat16):

Component	Formula	Memory Cost
Routing Buffer	L × d × K × 2 bytes	**~94 GB**
FFN Activations	2L × h × 2 bytes	**~98 GB**
Total Activation Memory per Layer		**~192 GB**

> A single MoE layer consumes nearly **200 GB** of activation memory — far exceeding the 80 GB HBM capacity of a modern H100 GPU.

Source 2: Memory Bandwidth Bottleneck

The Activation Penalty

Advanced non-linear activations significantly increase intermediate memory traffic during training.

- Kernels must materialize multiple intermediate tensors for the backward pass.
- These intermediates are written to and subsequently re-read from global memory.
- This repeated data movement generates massive memory traffic that scales with model dimensions.

Compute vs. Bandwidth on H100

Activation computation sits well below peak compute utilization.

H100 GPU UTILIZATION PROFILE

Required for Peak Compute	**~40 FLOPs/byte**
Element-Wise Intensity	**~1.25 FLOPs/byte**
Peak HBM3 Bandwidth	**3.35 TB/s**
Achievable Throughput	**< 5 TFLOPS**
Hardware Utilization	**< 1% of Peak**

MoEBlaze: Key Results

Millions

TOKENS SCALABILITY

Scales up to millions of tokens on the same hardware where baselines **run out of memory (OOM)**.

77%

LESS ACTIVATION MEMORY

Drastically reduces memory footprint by eliminating materialized routing buffers and fusing operations.

4-6x

FASTER TRAINING SPEED

Consistently achieves massive speedups over the most widely adopted open-source MoE training systems.

Zero

ACCURACY LOSS

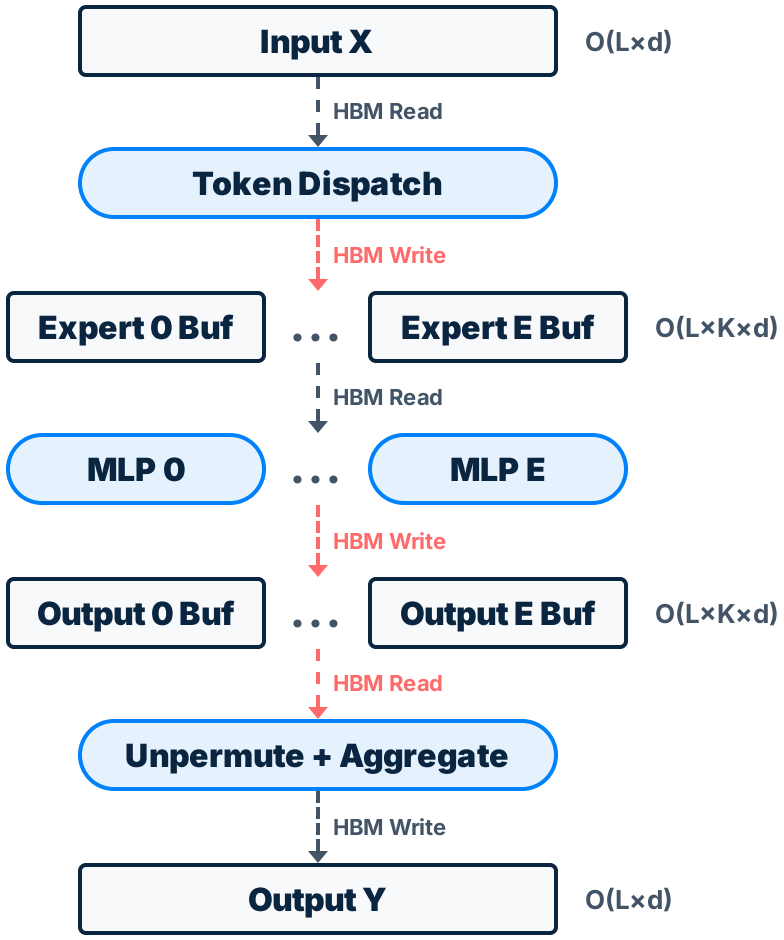
Mathematically equivalent computation. No token dropping, no padding required.

Conventional MoE vs. MoEBlaze

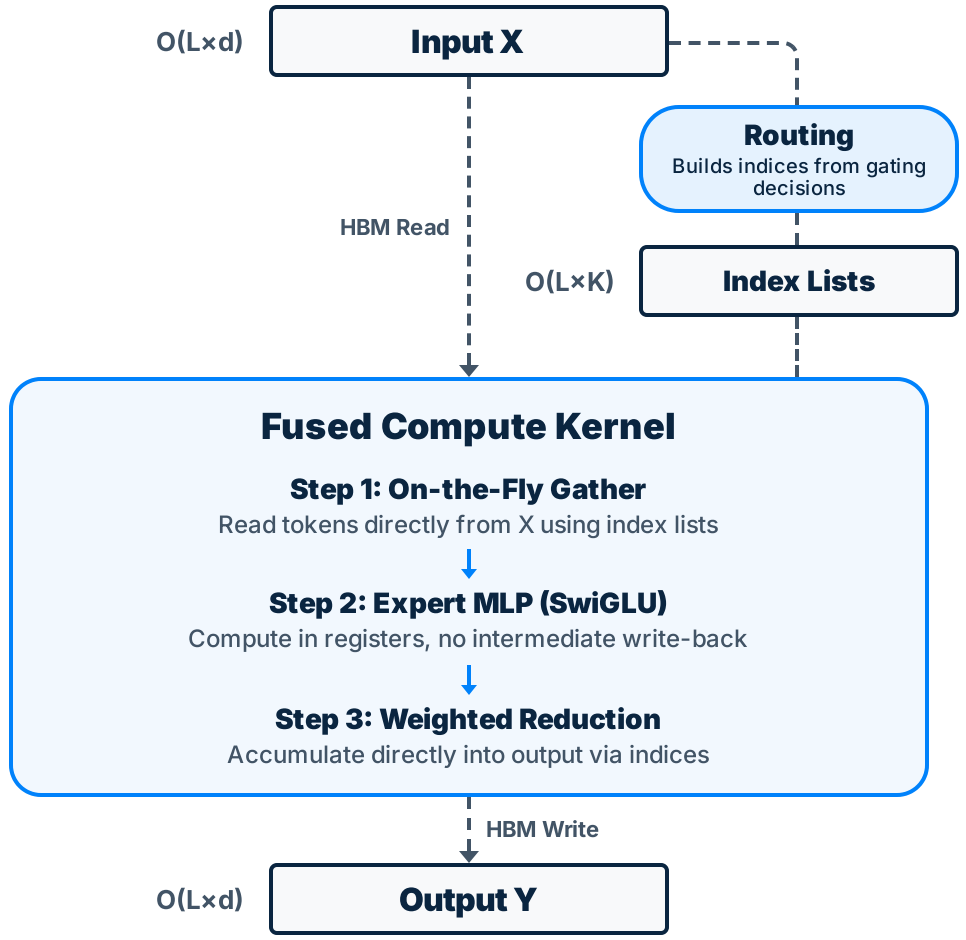
L = Sequence Length **h** = Intermediate Dim **E** = Total Experts
d = Hidden Dimension **K** = Top-K Experts

Data Tensor | Compute Kernel | Essential HBM R/W | Redundant HBM R/W

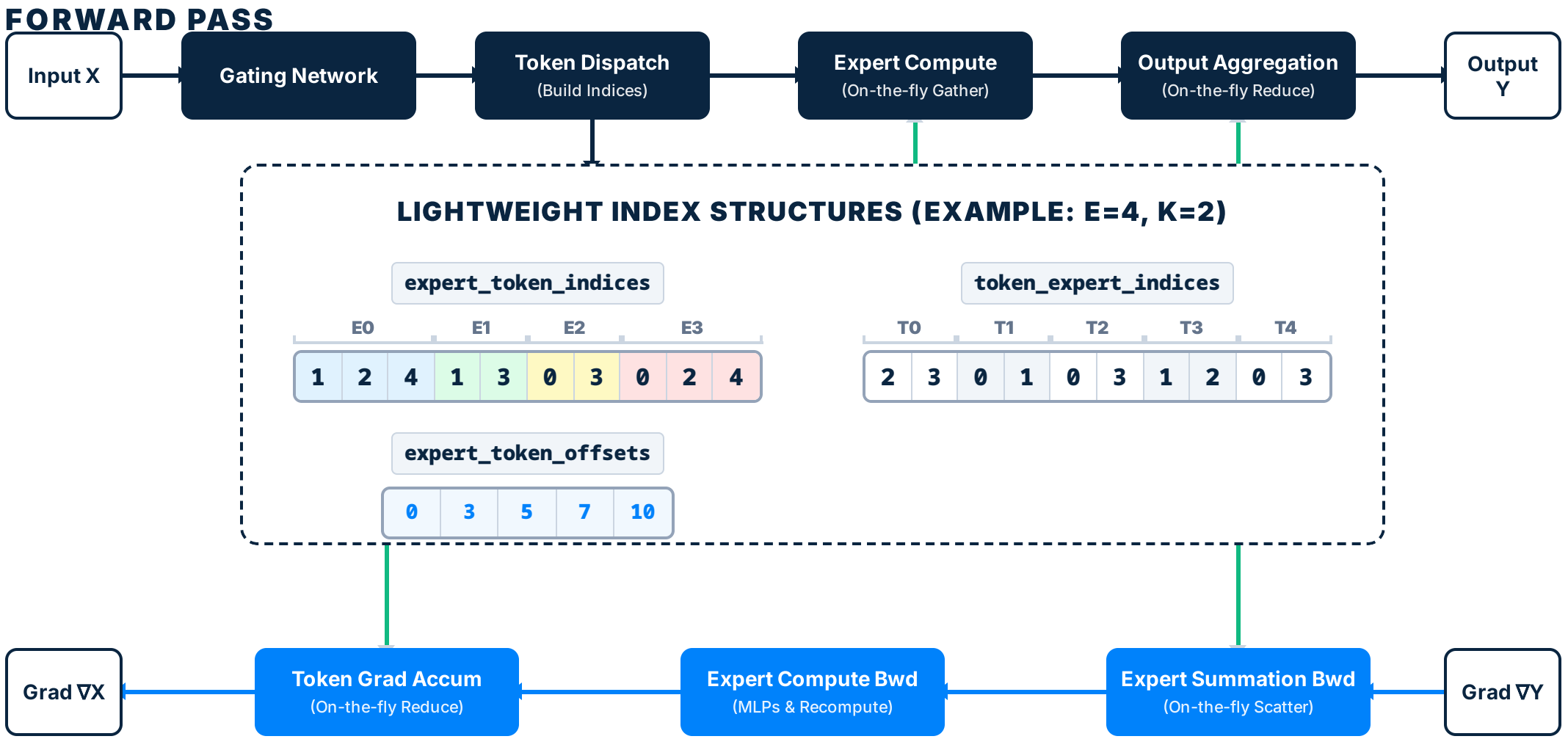
Conventional MoE



MoEBlaze



End-to-End Training Pipeline



GPU-Efficient Dispatching Algorithm

The Bottleneck: Sorting

- ✘ **Multi-pass radix sort** on GPUs requires several global-memory passes proportional to key width.
- ✘ Forces **O(L×k)** frequent global-memory passes, moving **data multiple times**.
- ✘ Results in high complexity, poor resource utilization, and **high kernel launch latencies**.
- ✘ Global ordering step limits fine-grained parallelism.

MoEBlaze: Lock-Free Parallel Algorithm

> 2x Speedup

1 Build Dense Token-Expert Map

Constructs a temporary **L × E dense bitmap** to encode top-k routing. Highly parallelizable on the GPU grid, with each warp handling a disjoint tile of token rows.

Zero intra-warp collisions

2 Compute Expert Lengths

Uses a custom kernel with **warp-level reductions** across the columns of the dense map to count tokens per expert. Computes lengths and derives offsets via prefix sum.

Efficient row-wise aggregation

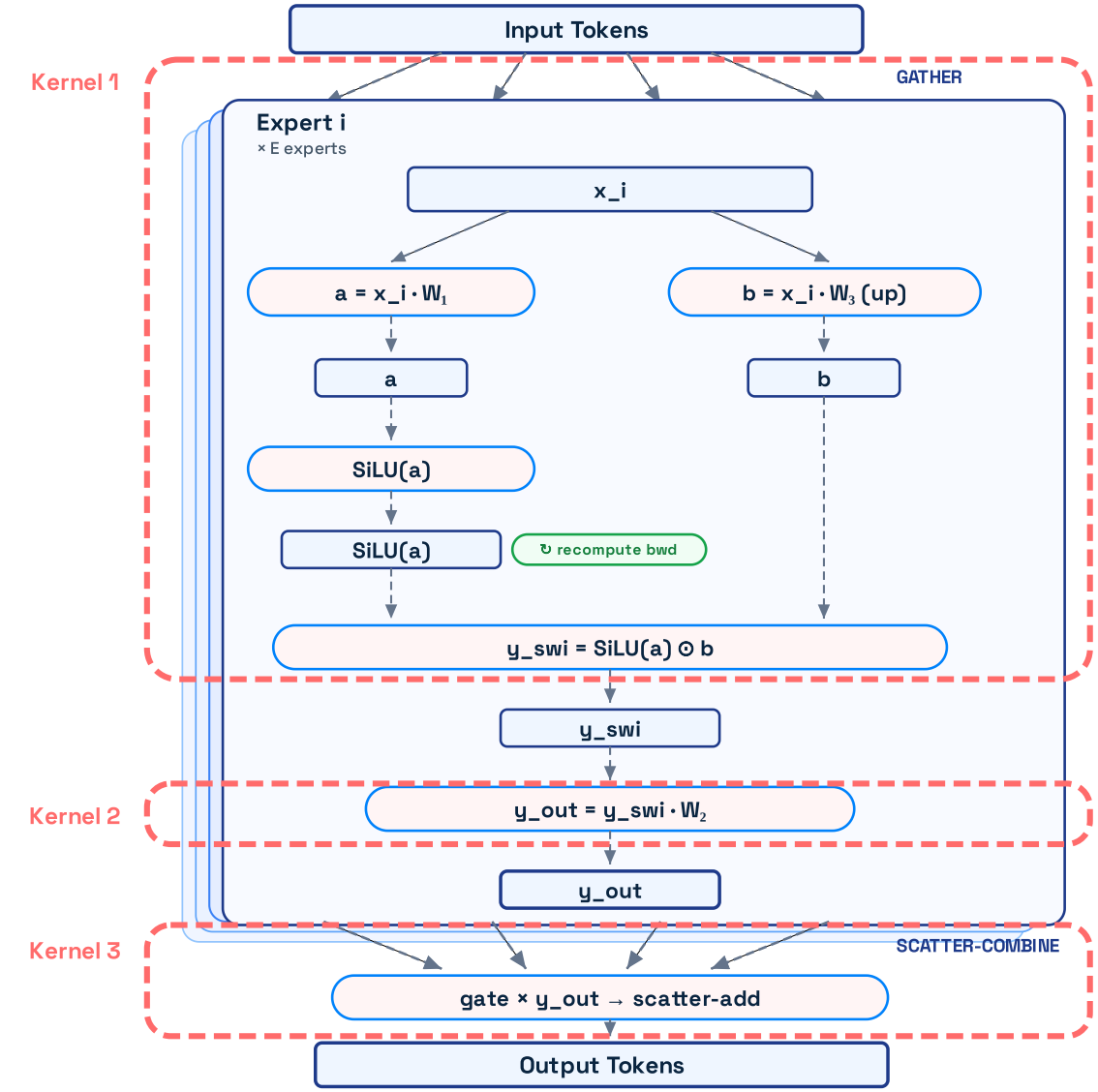
3 Route Indices to Gates

A two-phase tile-level scan generates a location map. A simple parallel kernel then reads from the dense map and **writes directly to final positions** in the concatenated indices array.

Full parallelism without atomics

Optimizing Expert Computation

3-level kernel fusion strategy



FUSION OPTIMIZATIONS

⚡ 1. Across Experts

Fuses all experts into a **single kernel**, eliminating the massive launch overhead of dispatching hundreds of individual expert kernels.

↓ Kernel launch overhead

𝔽 2. Within Expert

Computes SiLU(A) and element-wise multiplication entirely in **registers**. Intermediate results never touch global memory.

↓ HBM read/write traffic

♻ 3. Forward ↔ Backward

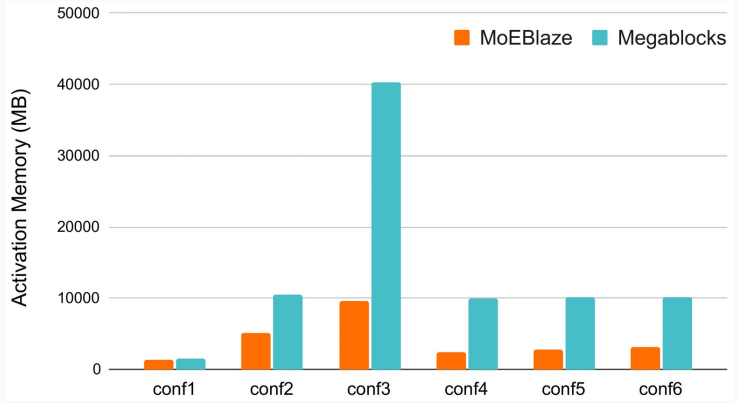
Drops transient SiLU(A) during forward pass and **recomputes it on-the-fly** during backward pass, saving significant activation memory.

↓ Activation memory

🎯 Together, these 3 levels reduce **HBM traffic** and **kernel overhead** — enabling near-compute-bound expert execution at scale.

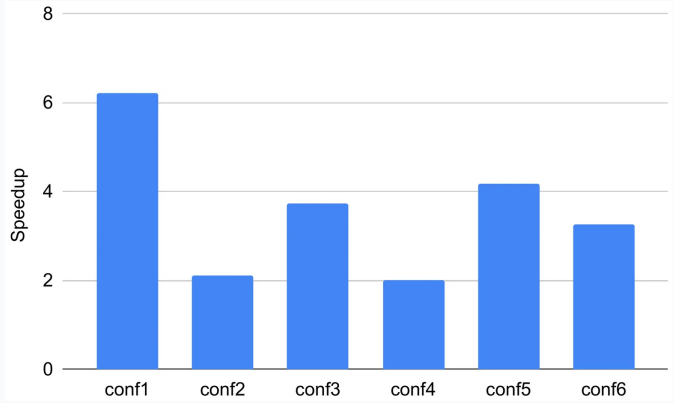
Performance and Scalability

Memory Efficiency (SwiGLU)



Consistently reduces activation memory footprint by >50%, avoiding OOM errors.

Training Speedup (SwiGLU)



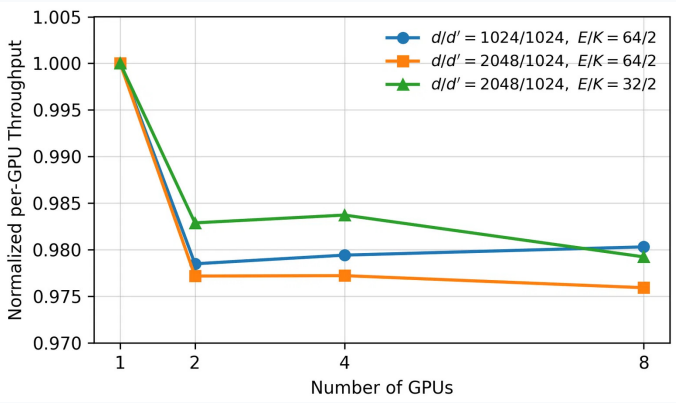
Achieves 1.9× to 6.2× end-to-end speedup across various MoE configurations.

Extreme-Scale Evaluation

Act.	Tokens	Time (ms)		Speedup	Mem. (MB)		Save (%)
		Baseline	Blaze		Baseline	Blaze	
SwiGLU	8k	1.8	1.0	1.9×	495	155	68.6%
SwiGLU	32k	5.9	2.3	2.6×	1931	477	75.3%
SwiGLU	128k	22.5	7.7	2.9×	7675	1766	77.0%
SwiGLU	512k	*OOM*	28.7	--	*OOM*	6918	--
SwiGLU	**2M**	*OOM*	**115.2**	--	*OOM*	**27528**	--

Scales stably up to 2M tokens, while baseline runs OOM beyond 512k tokens.

Multi-GPU Scalability (FSDP)



Maintains near-linear throughput scaling across multiple GPUs under FSDP.

Conclusion

MoEBlaze Breaks the Memory Wall

- **Massive Efficiency**

  Achieves **4-6x speedups** and **77% memory savings** by eliminating materialized routing buffers and fusing operations.

- **Extreme Scaling**

  Scales seamlessly to **Millions of tokens** on standard hardware without any accuracy loss.

Thank You

Questions & Discussion