Lecture 6: Attention and MoE Kernels

FlashAttention, Context Parallelism, and SonicMoE

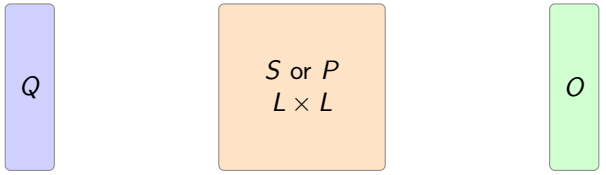
Fall 2026

FlashAttention

Attention: inputs and intermediates

Sequence length L, head width d: $Q, K, V \in \mathbb{R}^{L \times d}$.

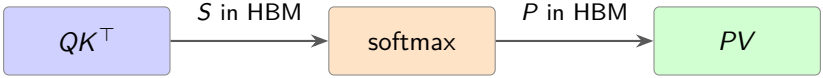
$$S = QK^\top/\sqrt{d}, \qquad P = \mathrm{softmax}(S), \qquad O = PV.$$



At $L = 32{,}768$, one BF16 score matrix takes **2 GiB per head**.
With 64 heads: **128 GiB** for one layer's scores alone.

Writing scores to HBM creates avoidable traffic

Materialized attention passes two intermediates through HBM:



- Each boundary writes and rereads an $L \times L$ tensor.
- Softmax has little arithmetic per byte moved.

Dao et al., FlashAttention, 2022.

A block average needs its weight

For one query, split the keys into two blocks. Toy unnormalized weights:

Block A

Weights: $1, 1$ Values: $0, 2$

Total weight: $z_A = 2$
Local output: $o_A = 1$

Block B

Weight: 6 Value: 4

Total weight: $z_B = 6$
Local output: $o_B = 4$

$$o = \frac{z_A o_A + z_B o_B}{z_A + z_B} = \frac{2 \cdot 1 + 6 \cdot 4}{8} = 3.25.$$

Averaging the two local outputs would incorrectly give 2.5.

Online softmax keeps a small, stable summary

For each query row, a block with scores s_j keeps:

$$\underbrace{m = \max_j s_j}_{\text{maximum}}, \qquad \underbrace{\ell = \sum_j e^{s_j - m}}_{\text{mass}}, \qquad \underbrace{u = \sum_j e^{s_j - m} v_j}_{\text{weighted sum}}.$$

Merge blocks a, b using $m = \max(m_a, m_b)$:

$$\ell = e^{m_a - m}\ell_a + e^{m_b - m}\ell_b,$$
$$u = e^{m_a - m}u_a + e^{m_b - m}u_b, \qquad o = u/\ell.$$

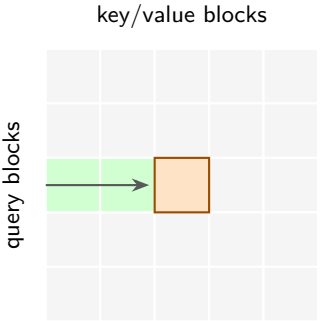
State per query

Two scalars and one d-dimensional vector, independent of context length.

Same attention, up to floating-point rounding.

FlashAttention, Algorithm 1. Rescaling preserves all previously processed keys.

FlashAttention streams tiles through on-chip memory



One query block visits all key blocks

1. Load a K, V tile.
2. Compute scores and update (m, ℓ, u) on-chip.
3. Discard the tile. Continue.

Write only final O and row statistics to HBM. Full S, P matrices never materialize.

Conceptual query-block schedule used by modern FlashAttention kernels (FA2).

Backward trades arithmetic for less saved memory

Forward saves

Q, K, V, output O, and one log-sum-exp per row:

$$\mathrm{LSE}_i = m_i + \log \ell_i.$$

Backward reconstructs tiles

Recompute scores and recover probabilities:

$$P_{ij} = \exp(S_{ij} - \mathrm{LSE}_i).$$

Then accumulate gradients tile by tile.

	Materialized attention	FlashAttention
Saved attention tensors	$O(L^2 + Ld)$	$O(Ld)$
Dense attention arithmetic	$O(L^2 d)$	$O(L^2 d)$

Per head. Dropout omitted. Recomputation adds FLOPs, so its time cost depends on the bottleneck.

FA2 and FA3 improve GPU utilization

	Bottleneck	Main idea
FA2	Too few workers and too much synchronization	Split query rows across blocks and warps. Reduce non-matmul work.
FA3	Data movement and softmax leave gaps between matmuls	Overlap asynchronous copies, matmuls, and softmax on Hopper.

FA3 also introduces an FP8 path with quantization error.

Dao, FlashAttention-2, 2023. Shah et al., FlashAttention-3, 2024.

Context Parallelism

Context parallelism splits one sequence across GPUs

When one sequence is too large, split its tokens across p GPUs.

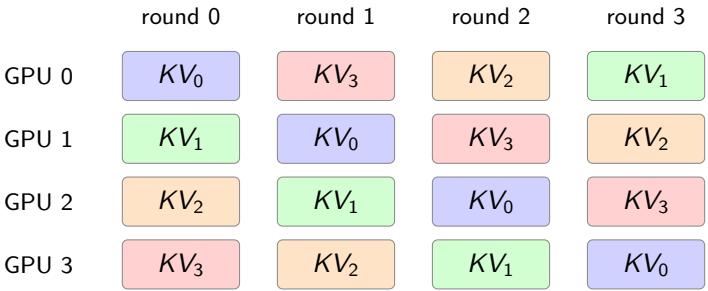
GPU 0	GPU 1	GPU 2	GPU 3
Q_r, K_r, V_r	Q_r, K_r, V_r	Q_r, K_r, V_r	Q_r, K_r, V_r
L/p tokens	L/p tokens	L/p tokens	L/p tokens

- Token-wise operations run on local tokens.
- Attention needs remote keys and values allowed by the mask.
- CP alone replicates weights. Training must combine their gradients.

Sequence activations shrink roughly p-fold, plus exchange buffers.

Megatron-Core, Context Parallelism. TP and EP can introduce their own communication.

Ring attention keeps queries local and rotates K, V



Q_r stays put.

Merge each partial into local output.

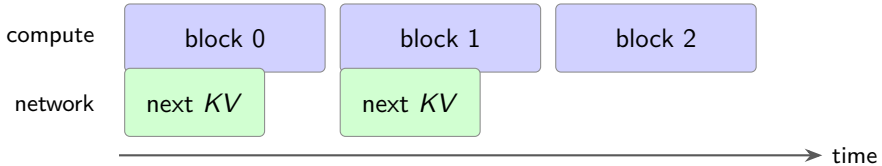
Four GPUs: **one local round + three neighbor exchanges**.
Each GPU sends to the next rank and receives from the previous rank.

Liu et al., Ring Attention, 2023. Full attention shown. Causal masks skip invalid query-key pairs.

Long local chunks make overlap easier

One full ring block pair. Local length $c = L/p$, model width $D = n_q d$.

$$t_{\text{compute}} \approx \frac{4c^2 D}{F_{\text{eff}}}, \qquad t_{\text{comm}} \approx \alpha + \frac{2c\, n_{kv} d\, b}{BW_{\text{eff}}}.$$



Scaling limit

At fixed L, doubling p quarters compute per round but only halves its payload.

n_q, n_{kv}: query/KV heads. b: bytes/element. $F_{\text{eff}}, BW_{\text{eff}}$: achieved FLOPs/s and link bytes/s. One sequence, forward only. Ring Attention, 2023.

Causal attention needs balanced token placement

Contiguous ownership

Work: $3, 7, 11, 15$

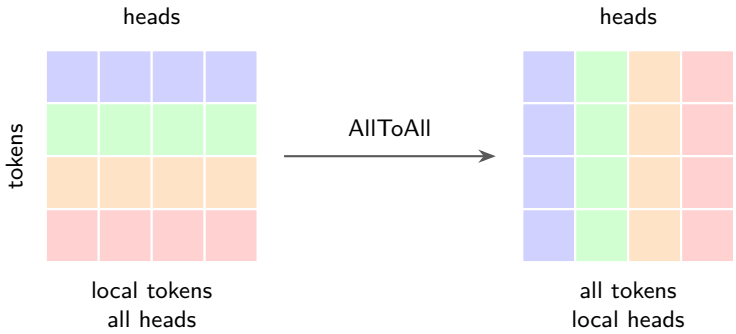
Pair early and late tokens

Work: $9, 9, 9, 9$

Eight-token toy example. Color identifies the owning GPU.
Rows are queries, columns are keys. Future positions are masked.

Megatron-Core CP uses paired sequence chunks for causal load balance. Global positions stay unchanged.

Ulysses trades sequence shards for head shards



Run full-sequence FlashAttention on local heads. A second AllToAll restores token ownership.

DeepSpeed-Ulysses. Two logical layout exchanges in forward, two in backward. Color = rank.

The exchange strategy changes the trade-off

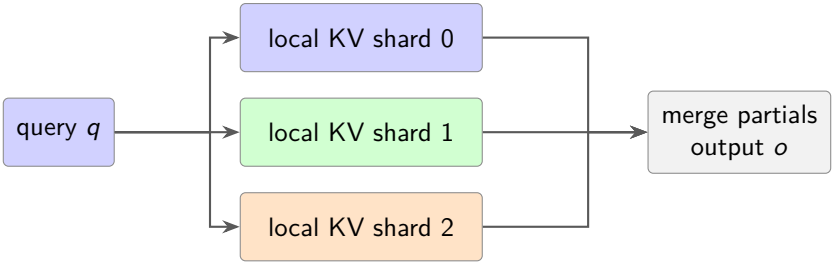
Method	Constraint to check
Ring	Enough local compute to overlap each transfer
Ulysses	Head divisibility, especially few KV heads under GQA
AllGather K, V	Simpler local attention, but every rank needs full gathered K, V

Terminology: Megatron SP only shards selected activations within TP. CP also partitions attention across the sequence.

Megatron-Core CP; DeepSpeed-Ulysses. GQA = grouped-query attention. Terminology varies by framework.

Decode can keep the KV cache sharded

One decode query per request:



Broadcast the query, scan local KV, then combine small summaries.

Benefit: more memory capacity and aggregate HBM bandwidth.
Cost: communication and synchronization on every decode step.

Conceptual KV-parallel decode. See Bhatia et al., Helix Parallelism, 2025. Exchange schedules vary by implementation.

MoE Kernels: SonicMoE Case Study

Expert granularity and sparsity: the kernel costs

Let T = tokens, k = selected experts, f = expert hidden width.

Finer experts

Halve f, double k.
Activated FFN FLOPs stay roughly fixed: $\propto TkDf$.

Expanded $Tk \times D$ buffers double.

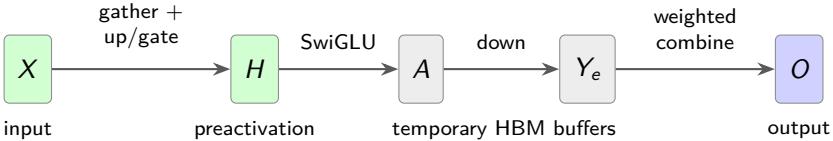
Sparser routing

Increase total experts E at fixed k.
Average tokens per expert fall: $\bar{M} = Tk/E$.

Small expert batches leave partly filled GEMM tiles.

Grouped GEMM runs several experts' matrix multiplies in one launch. Guo et al., SonicMoE, ICLR 2026.

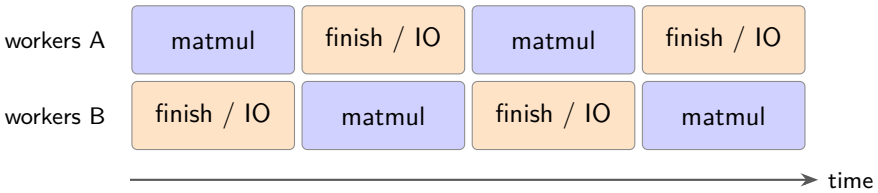
Saved activations versus temporary buffers



- Fuse input gathering into the matrix-multiply loads.
- Reorder backward to use X, H and routing metadata.
- Release A, Y_e after forward use. Recompute SwiGLU in backward.

SonicMoE, section 3. Green = saved activations. SwiGLU is recomputed, GEMMs are not.

Overlapping grouped GEMM with IO



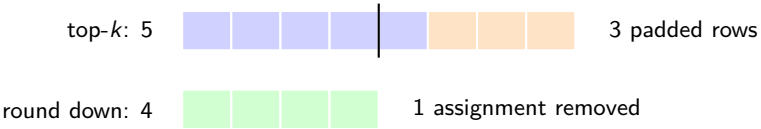
The **epilogue** finishes a tile: activation functions, reductions, and stores.

The SonicMoE kernels overlap this work with another tile's matmul, as in FA3.

SonicMoE, section 4. Schematic Hopper ping-pong schedule. Exact timings and hardware schedules vary.

Token rounding changes routing to fill whole tiles

Toy example: tile size $= 4$ token-expert assignments.



Counts may round **down or up** to tile multiples, guided by router scores.

Optional: compare with unchanged top-k routing and evaluate model quality.

SonicMoE, section 5. The kernel optimizations also work with unchanged top-k routing.

Summary

- FlashAttention tiles Q, K, V through on-chip memory with online softmax; S and P are never written to HBM. Activation memory drops from $O(L^2)$ to $O(Ld)$; FLOPs are unchanged.
- CP shards the sequence across GPUs. Ring attention rotates K, V and overlaps the transfer with compute; Ulysses trades sequence shards for head shards. Causal masks need balanced token placement.
- SonicMoE overlaps grouped GEMM with IO, recomputes SwiGLU instead of saving expert outputs, and rounds token counts to fill whole tiles.
- Fusion, rescaling, and backward reordering preserve results up to floating-point rounding. Token rounding changes routing, so it changes the computation.