Attention and GEMM Kernels

Deep Dive into SGLang — Chapter 27

Yin Yang

Foundation Books Project

Where this chapter sits

This is the chapter that turns "pick an attention backend" and "multiply a projection" into one concrete device launch.

- Volume I supplied live request rows, cache visibility, masks, and output ownership.
- Here the runtime chooses a concrete tile and storage plan, records page tables and scale records, then proves the physical plan still implements the scheduled operator.
- Two launches carry the whole chapter: a decode-attention row and a projection GEMM.

Goal: read a fast kernel and say which request row owns the output, which metadata gives the bytes meaning, and what baseline makes the number comparable.

Roadmap

The launch-level problem

Serving kernel shape

Attention: online softmax

GEMM: tiling and scale records

Correctness and measurement

The launch-level problem

Two launches, one rule

Half	Request rows	Kernel-facing view
Decode-heavy	A, B: one new row each	Attention over long KV histories: lengths, masks, pages / KV slots, output rows.
Prefill-heavy	C: 128 suffix rows	Projection GEMM $K = N = 4096$: padding, packed weights, scale layout.

Both launches can be fast and still wrong: the attention path can read the wrong KV row; the GEMM path can apply the wrong FP8 scale block or return padded rows as tokens.

Tile order, split work, padding, and packing may change the physical launch shape — the outputs must still attach to the scheduled request rows.

Serving kernel shape

The serving kernel shape names more than tensor extents

Definition (Serving kernel shape)
$S_t = (o_t, \mathcal{B}_t, \mathbf{s}_t, \mathbf{g}_t, \tau_{\text{tile}}, \Delta_{\text{dtype}}, \kappa)$: the logical operator o_t, scheduled rows $\mathcal{B}_t$, sequence state $\mathbf{s}_t$, geometry $\mathbf{g}_t$, tile policy τ_{tile}, dtype/accumulation Δ_{dtype}, and the layout bundle κ.

- $\mathbf{g}_t = (H_Q, H_{\text{KV}}, d_h)$ for attention, or (M, K, N) for GEMM.
- κ collects cache indices, page tables, scale layout, padding, workspace, and output slicing.
- Arithmetic intensity $I(S_t) = \text{FLOPs}(S_t)/\text{bytes}(S_t)$ gives this instance a cost, not the abstract operator.

Same output extent, different κ: the launch is a different operation.

Attention: online softmax

Attention state is three registers per query row

For one query row, keep a running max m, a running normalizer ℓ, and an unnormalized accumulator u. When score tile $s_j = qk_j$ and value tile v_j arrive:

$$m^+ = \max(m, \max_j s_j), \qquad \ell^+ = e^{m-m^+}\ell + \textstyle\sum_j e^{s_j - m^+},$$

$$u^+ = e^{m-m^+}u + \textstyle\sum_j e^{s_j - m^+} v_j, \qquad \text{output} = u^+/\ell^+.$$

This is the mechanism behind "maintains softmax state across tiles": the kernel never materializes the full score matrix, yet the registers still belong to *one* query row until the final write.

GEMM: tiling and scale records

A GEMM launch is a tile loop plus a legal launch record

For $C = AB$, the kernel decomposes C into CTA tiles $M_b \times N_b$, walks K in K_b chunks, stages A, B fragments on-chip, and lets warps issue MMA into register accumulators until the epilogue.

- Dense work $\approx 2MKN$ FLOPs; the byte term includes scale blocks and packed weights.
- A *quantized launch record* adds packed weights, scale/zero-point layout, architecture dispatch, padding, and output slice.
- Blockwise FP8 groups activation scales over 128-wide K and weight scales over 128×128 tiles.

The bytes change; the requested matrix product must not.

Correctness and measurement

Launch correctness: reorder tiles, never the operator

Invariant (Kernel step-alignment)
A kernel may reorder tiles and memory accesses, but it must preserve the logical attention mask, sequence lengths, and matrix-product semantics.

Shape-compatible bugs that still "pass": wrong KV row, wrong FP8 scale block, stale graph-bucket row, missing sink or logit cap, invalid sparse compaction.

Semantic modifiers — sinks, softcap, temperature — live in the launch record. An attention sink absorbs softmax mass through an extra virtual logit $\exp(s_h)$ in the denominator, with no value contribution.

A measurement counts only after the reference check

- Change one launch choice — split policy, kernel family, packing, dtype — and hold the serving case, hardware, and window fixed.
- Attention evidence proves the same rows and visible KV history reached the kernel; GEMM evidence proves the same M, N, K survived dtype and packing.
- A roofline lower bound: $t_{\text{step}} \gtrsim \max(R_{\text{bytes}}/B_{\text{mem}}, W_{\text{flop}}/F_{\text{comp}})$ plus launch, collective, copy/pad terms.

A faster path that changed masks, scale blocks, or output slicing is a correctness failure, even when the metric improves.

What to carry forward

1. A kernel choice is a **serving decision**, not an isolated microbenchmark.
2. Layout metadata is **semantic state**: page tables, index lists, scale blocks, and output slices are the evidence bytes still denote the operator.
3. Attention's risk is **reading the wrong history or writing the wrong row**; GEMM's risk is **packing/scale/padding changing the product**.
4. **Online softmax** and the **CTA tile loop** explain why memory order may change while semantics do not.
5. A measured row counts only after **step-alignment and metadata accounting** hold.

Next: from the launch-level problem to the vocabulary of shape, layout, and cost that makes one attention or GEMM launch comparable.