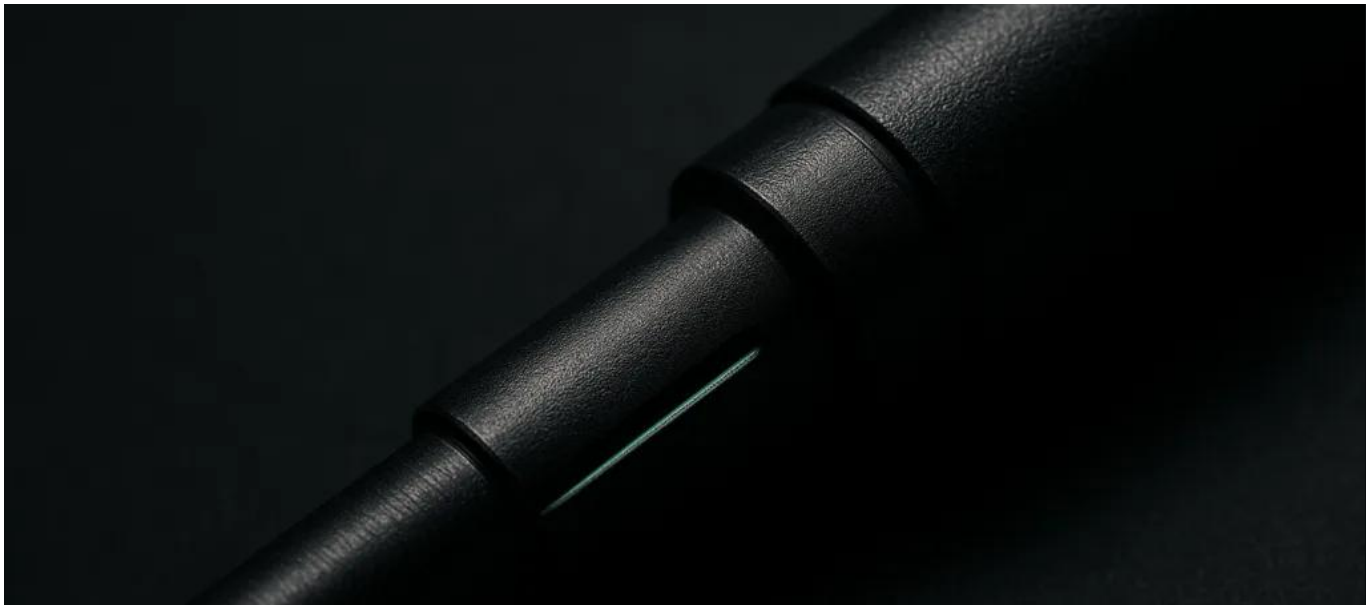

SGLang 0.5.19 in Practice: Beam Search, a Faster Cold Start, and What Still Breaks

SGLang's 0.5.19 headline is beam search, and it will not run on a disaggregated deployment with speculative decoding. The change I would actually deploy for shipped two weeks earlier, in 0.5.18.

SGLang shipped v0.5.19 on 5 September 2026, per the sgl-project releases page: 786 merged pull requests from roughly 214 contributors. The headline is beam search, exposed as a `beam_width` parameter that returns the n best sequences instead of one sample. It does not work with speculative decoding, prefill/decode disaggregation, DP attention, or HiCache, according to the v0.5.19 notes. If your production config is a disaggregated DeepSeek-class deployment with EAGLE-style drafting, beam search is not available to you in this version. Read it as a correctness feature for eval and reranking pipelines running on plain, single-node configs.

The boring install facts: Apache-2.0, maintained by the sgl-project GitHub org with LMSYS-affiliated researchers and engineers, `pip install sglang`, CUDA 12.x for H100/B200 and

recent ROCm for MI300X/MI355X. There is no commercial tier gating any of the features below.

The cold start fix came from v0.5.18, not the new release

v0.5.18, around 22 August 2026, added overlapped checkpoint staging via `–startup-weight-load-mode overlap`, which runs weight loading and CUDA graph capture concurrently instead of serially. That is the flag I would reach for first.

Cold start is the number that decides whether autoscaling self-hosted inference is a real strategy or a spreadsheet fiction. If a replica takes over a minute to answer its first request, you either overprovision or you eat the latency cliff on every scale-out event. This is also cumulative with v0.5.15's Breakable CUDA Graph work, merged as the default capture path on CUDA on 2 July 2026, which reported build speed 3.8 to 5.2× faster than the backend it replaced, replay 17% faster, and on a CoreWeave DeepSeek V4 test throughput up 11.80% with median time-per-output-token down 13.27%.

Lean attention is on by default on MI300X and MI355X

The persistent Lean attention kernel ships enabled by default on MI300X and MI355X, and can be turned off with `SGLANG_DISABLE_LEAN_ATTENTION=1`, per the AI Toolchain newsletter. I have not run it on my own MI355X shapes, so I am taking the default-on decision as the evidence rather than any number.

That a vendor-specific kernel is on by default, with an opt-out env var rather than an opt-in flag, tells you the maintainers consider it stable enough to own the regression risk. **That is the signal worth reading here.** A maintainer who makes a new kernel the path everyone hits, with a documented kill switch, is betting their issue tracker on it. (Opinion, obviously. Yours may differ if you have been burned by a default-on kernel before.)

The radix cache and DeepEP v2 are the two config diffs

The unified radix tree is now the default prefix cache for every model, replacing the separate per-model implementations. For anyone who has debugged why cache hit rates

differed between two models on the same box, this removes a whole class of surprises. It also means one code path now carries the blame for all of them, which cuts both ways in month one of a release.

The DeepEP v2 fixed-size ElasticBuffer MoE backend, selectable with `–moe-a2a-backend deepep_v2`, enables MoE decode under CUDA graphs across nodes. That combination (multi-node MoE, graph-captured decode) was the awkward gap in large mixture-of-experts serving, and it is the change I would test first on any DeepSeek- or Qwen-MoE deployment.

Then there is `–enable-layernorm-sp`, which has each tensor-parallel rank normalise only its own share of prefill tokens. Reported gain: 3.5% of prefill time removed on Qwen3-8B on H100, 5.6% on B200. Small, honestly measured, opt-in. I like flags like this more than headline features, because the size of the claim matches the size of the change.

WHAT WORKS

- Overlapped checkpoint staging runs weight loading and graph capture concurrently.
- Lean attention default-on for MI300X/MI355X, opt-out via env var.
- Multi-node MoE decode under CUDA graphs via `deepep_v2`.
- Apache-2.0, no paid tier on any of it.

ROUGH EDGES

- Beam search incompatible with spec decoding, disaggregation, DP attention, HiCache.
- The layernorm win is 3.5% to 5.6%, on two specific models and two specific GPUs.
- Unified radix cache concentrates all prefix-cache risk in one new default path.
- 786 PRs in one release is a lot of surface to have regressed.

vLLM has the bigger ecosystem, SGLang has the faster cadence

vLLM shipped v0.29.0 on 9 September 2026 with 411 commits from 212 contributors, also Apache-2.0, at roughly 85.6k GitHub stars. That star count is a hiring and documentation signal more than a technical one: it is easier to find engineers who have run vLLM, and easier to find an answer when something breaks at 2am. SGLang's 786 PRs in a single release read as faster movement at the cost of more churn per upgrade.

My take: if you are serving a MoE model across nodes, or running MI300X/MI355X hardware, SGLang 0.5.19 is worth the migration cost now. If you are on a single-node dense model on H100 and your cold start is not in your error budget, skip this one and take 0.5.18's staging flag plus whatever your current engine gives you. And if you came for beam search, check your feature matrix against those four incompatibilities before you write the ticket, because the same pattern shows up in every young serving feature I have reviewed this year, including the ones I wrote up in the StackOne Defender review: the headline lands before the integration surface does.

The measurement I would run before committing anything is your own prefill/decode ratio against the layernorm and radix-tree changes, because a 3.5% prefill win means nothing on a workload that is 90% decode. If you want a second pair of eyes on that number for your actual traffic mix, send me your serving config and I will tell you which of these flags will move it.

WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →