# PR #31324 完整报告

sgl-project/sglang

[AMD] [GLM5] Skip DSA decode indexer when kv_len <= index_topk (dense k-only fast path)

合并时间：2026-08-17 07:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31324

## 执行摘要

- 一句话：DSA decode 跳过 indexer 的 k-only 快路径，吞吐提升约 5%
- 推荐动作：值得精读。核心看点有三：一是 CUDA graph 捕获期分支冻结这一深层约束下的解决思路——不在 capture 期按运行时 seq_len 分支，而是捕获 dense/sparse 两张图并在 replay 时由 host 分发；二是 _resolve_dsa_variant 优先复用 host 侧 seq_lens_cpu 镜像、避免每步 d2h 同步的性能细节；三是硬件 gating 的取舍——明确承认 CUDA 未验证并刻意保持原行为，值得作为跨平台共享代码变更的参考样本。若团队后续要支持 CUDA DSA 模型，建议先补齐 CUDA 上 DSA 模型的 decode 验证再摘除 is_hip() 门控。

## 功能与动机

PR body 明确指出：当请求的 kv_len <= index_topk 时，top-k 会选中所有有效位置，因此 indexer 的 logits GEMM、paged_mqa_logits 与 top-k 选择全是浪费工作，应增加一条跳过 indexer、直接生成 identity index 的 k-only 快路径。提交历史还记录了正确性教训：早期 spike 用 arange() 直接合成索引，在 SGLANG_DSA_FUSE_TOPK 下是错误的——索引必须是 physical page-slot 而非逻辑位置，导致 GSM8K 掉 0.02，因此最终改用 dummy-logits 走 topk_transform 得到正确索引。

## 实现拆解

实现按 4 步展开：

1. 扩展 DSA indexer 的 k-only 快路径到 decode（layers/attention/dsa/dsa_indexer.py）：_should_skip_logits_computation 从仅支持 extend 扩到 decode/idle，并按 capture 与 eager 两分支分流——capture 模式下不能按运行时 seq_len 分支（分支会被冻结且 host 同步会破坏捕获），改由进程级 _capture_dsa_variant 信号决定；eager 模式则每步 host 同步检查 max_kv_len <= index_topk。_forward_cuda_k_only 的断言同步放宽到接受 decode 模式，并保留 num_tokens（图内 padding 裁剪）与 topk_result（图内固定地址回填）两个图契约参数。
2. 新增进程级 capture 变体信号（model_executor/runner_utils/capture_mode.py）：仿照既有 _capture_lora_variant 增加 _capture_dsa_variant 全局变量及 get_capture_dsa_variant / _set_capture_dsa_variant，取值 "dense" / "sparse" / None（None 表示非双图捕获，indexer 默认烘焙完整 indexer 路径，对任意 kv_len 都正确）。

3. decode runner 双图捕获与 host 分发（model_executor/runner/decode_cuda_graph_runner.py）：__init__ 中当 is_hip() and is_deepseek_dsa(hf_config) 时从 HF config 读取 index_topk 并自动置 dsa_dual_graph = True（约增加 52 张图、2 倍捕获时间，代码注释中已声明）；_capture_one_stream 对每个 bs 桶先捕获 dense（峰值更小）再捕获 sparse；load_batch 时由 _resolve_dsa_variant 取批内 max_kv_len 与 dsa_index_topk 比较选择回放图，任一请求超长则整批退回 sparse；_make_graph_key 把 dsa_variant 纳入图缓存键。
4. ShapeKey 数据契约扩展（model_executor/runner/shape_key.py）：ShapeKey 新增 dsa_variant: Optional[str] 字段，并与已有 variant_label（LoRA）正交组合，保证 dense/sparse 与 lora/nolora 图在缓存键层面互不串扰。

配套与测试：本 PR 未新增测试文件；准确性（GSM8K 0.941 vs 基线 0.922）与性能（MI355X TP4、GLM-5.2-MXFP4、tilelang DSA 后端）均来自 AMD nightly/ 手工验证。过程中还修复了 --mtp 下 EAGLEDraftCudaGraphRunner 复用 capture() 时因未执行本类 __init__ 而缺失 dsa_dual_graph 属性的 AttributeError，改用 getattr(..., False) 默认值并在 dsa_variant is None 时不传额外参数，保证子类覆盖的 capture_one_shape 签名兼容。

关键文件：

- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 解码图引擎；类别 source；类型 core-logic；符号 _make_graph_key, _resolve_dsa_variant, _capture_one_stream, load_batch）：本 PR 的核心载体：新增 dsa_dual_graph 自动启用逻辑、dense/sparse 双图捕获循环、host 端 _resolve_dsa_variant 分发，以及 _make_graph_key 的 dsa_variant 维度扩展；同时处理了 EAGLE 子类复用 capture() 的兼容。
- python/sglang/srt/layers/attention/dsa/dsa_indexer.py（模块 DSA 索引器；类别 source；类型 core-logic；符号 _should_skip_logits_computation, _forward_cuda_k_only）：k-only 快路径的语义核心：_should_skip_logits_computation 扩展到 decode 并区分 capture/eager 两分支，_forward_cuda_k_only 断言放宽并保持图契约参数，是整个优化的正确性根基。
- python/sglang/srt/model_executor/runner_utils/capture_mode.py（模块 捕获模式；类别 source；类型 data-contract；符号 get_capture_dsa_variant, _set_capture_dsa_variant）：新增进程级 _capture_dsa_variant 信号，仿照 LoRA 双图机制，供 indexer 在 capture 期读取当前正在捕获的图变体，是 capture 期选路的关键数据通道。
- python/sglang/srt/model_executor/runner/shape_key.py（模块 图键标识；类别 source；类型 data-contract；符号 ShapeKey）：ShapeKey 数据契约扩展：新增 dsa_variant 字段并与 variant_label（LoRA）正交组合，保证 dense/sparse 图在缓存键层面不串扰，是双图能共存的键基础。

关键符号：_resolve_dsa_variant, _make_graph_key, _should_skip_logits_computation, _forward_cuda_k_only, get_capture_dsa_variant, _set_capture_dsa_variant, _capture_one_stream

## 关键源码片段

## python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

本 PR 的核心载体：新增 dsa_dual_graph 自动启用逻辑、dense/sparse 双图捕获循环、host 端 _resolve_dsa_variant 分发，以及 _make_graph_key 的 dsa_variant 维度扩展；同时处理了 EAGLE 子类复用 capture() 的兼容。

```
# decode_cuda_graph_runner.py —— DSA 双图自动启用 + replay 时 host 分发的核心
# 设计要点：CUDA 图捕获期分支会被冻结，不能在 capture 时按运行时 seq_len 选路，
# 所以改为每个 bs 桶捕获 dense（k-only）与 sparse（完整 indexer）两张图，
# 在 replay 时由 host 侧按批内最大 kv_len 决定回放哪一张。

def __init__(self, model_runner, ...):
    # --- DSA dense-decode dual-graph ---
    # 仅 HIP（AMD）启用：k-only 快路径只在 MI355X 上验证过，
    # CUDA 上刻意保持原 sparse-only 行为，避免静默改变 DeepSeek-V3.2
    # 等 DSA 模型在 CUDA 的 decode 路径。
    self.dsa_dual_graph = False
    self.dsa_index_topk: Optional[int] = None
    from sglang.srt.configs.model_config import get_dsa_index_topk, is_deepseek_dsa

    hf_config = model_runner.model_config.hf_config
    if is_hip() and is_deepseek_dsa(hf_config):
        self.dsa_index_topk = get_dsa_index_topk(hf_config)
        self.dsa_dual_graph = True
        logger.info(
            "[dense-decode] DSA dual-graph enabled: capturing "
            "dense (k-only) + sparse (full indexer) decode graphs; "
            "dispatch on max_kv_len vs index_topk=%d.",
            self.dsa_index_topk,
        )

def _resolve_dsa_variant(self, forward_batch: ForwardBatch) -> Optional[str]:
    """replay 时的 host 分发：取批内 max_kv_len 与 index_topk 比较。
    只要批内有一个请求 kv_len > index_topk，dense（k-only）图对它就是错的，
    所以整批退回对任意长度都正确的 sparse 图。返回 None 表示双图未启用。"""
    if not getattr(self, "dsa_dual_graph", False):
        return None
    seq_lens_cpu = getattr(forward_batch, "seq_lens_cpu", None)
    if seq_lens_cpu is not None and seq_lens_cpu.numel() > 0:
        # 优先走 host 侧镜像（普通 decode 增量维护），避免每步 d2h 同步
        max_kv_len = int(seq_lens_cpu.max().item())
    elif forward_batch.seq_lens is not None and forward_batch.seq_lens.numel() > 0:
        # 兜底：单次标量归约 d2h，每步一次，开销可接受
        max_kv_len = int(forward_batch.seq_lens.max().item())
    else:
        # 拿不到长度信息时，用对任意 kv_len 都正确的 sparse 图
        return "sparse"
    return "dense" if max_kv_len <= self.dsa_index_topk else "sparse"
```

## python/sglang/srt/layers/attention/dsa/dsa_indexer.py

k-only 快路径的语义核心：_should_skip_logits_computation 扩展到 decode 并区分 capture/eager 两分支，_forward_cuda_k_only 断言放宽并保持图契约参数，是整个优化的正确性根基。

```
# dsa_indexer.py —— decode k-only 快路径的选路逻辑（capture 与 eager 两分支）
# 正确性前提：kv_len <= index_topk 时 top-k 会选中全部有效位置，
# 因此 logits GEMM + paged_mqa_logits + top-k 都是浪费；用 dummy-logits
# 走 topk_transform 即可得到正确的 physical page-slot 索引（identity 索引）。

def _should_skip_logits_computation(self, forward_batch: ForwardBatch) -> bool:
    fb = forward_batch

    # Prefill/extend：所有平台共有的老快路径，host 同步读 seq_lens_cpu 即可
    if fb.forward_mode.is_extend_without_speculative():
        if fb.seq_lens_cpu is None or fb.seq_lens_cpu.numel() == 0:
            return False
        return int(fb.seq_lens_cpu.max().item()) <= self.index_topk

    # Decode/idle：本 PR 新增，暂时仅限 ROCm（CUDA 未验证，decode 永不跳过）
    if fb.forward_mode.is_decode_or_idle():
        if not _is_hip:
            return False
        if get_is_capture_mode():
            # 关键约束：CUDA 图捕获时分支被冻结，不能在 capture 期按运行时
            # seq_len 分支（host 同步也会破坏捕获）。改由 runner 正在捕获
            # 哪个图变体（dense/sparse）来驱动选路。
            from sglang.srt.model_executor.runner_utils.capture_mode import (
                get_capture_dsa_variant,
            )
            variant = get_capture_dsa_variant()
            if variant == "dense":
                return True
            if variant == "sparse":
                return False
            # 无双图捕获信号：默认走对任意 kv_len 都正确的完整 indexer 路径
            return False
        # Eager decode：每步 host 同步检查一次，两种长度都正确
        if fb.seq_lens_cpu is not None and fb.seq_lens_cpu.numel() > 0:
            max_kv_len = int(fb.seq_lens_cpu.max().item())
        elif fb.seq_lens is not None and fb.seq_lens.numel() > 0:
            max_kv_len = int(fb.seq_lens.max().item())
        else:
            return False
        return max_kv_len <= self.index_topk

    return False
```

# 评论区精华

评审中的核心交锋集中在三处：

1. clintg6 的 CHANGES_REQUESTED：实测 MI355X TP8 下约 3% 吞吐提升后，要求删除三个调试 /opt-in 环境变量——SGLANG_KONLY_DEBUG（残留的调试日志开关）、SGLANG_DSA_DECODE_DUAL_GRAPH（用户不应手动开启双图）、SGLANG_DSA_DECODE_DENSE_GRAPH（无条件绕过 indexer 的危险调试覆盖）。结论：全部移除，改用 dsa_dual_graph = dsa_index_topk is not None 自动启用，clintg6 复测后 APPROVED。
2. sogalin 的 CUDA 质疑："Have you tried the change on CUDA? It is common code change in SGL." 作者承认未在 CUDA 验证，随后加 is_hip() 保护，且后续提交发现 eager decode 分支也存在独立的未验证路径，一并补上 _is_hip 门控，确保 CUDA 上 DSA 模型 decode 永不走 k-only。sogalin 回复 "LGTM, having is_hip to protect now"。
3. amd-bot 反复强调的 CI 覆盖缺口：双图仅在 is_hip() and is_deepseek_dsa() 下启用，而所有 DSA/GLM-5 端到端测试注册为 nightly=True，PR CI 永远跑不到改动代码，绿跑只能证明模块可导入、非 DSA 路径未破坏。结论：合入前需在 AMD 上手动跑 GLM5/DeepSeek-V3.2 decode eval，clintg6 已完成实测。

- 移除三个调试 /opt-in 环境变量，DSA 模型自动启用双图 (design): 三个环境变量全部删除，改为 dsa_dual_graph = dsa_index_topk is not None 基于 HF config 自动启用；clintg6 复测后 APPROVED。
- CUDA 平台安全性：共享代码未在 CUDA 验证 (question): 加 is_hip() 门控，且后续提交发现 eager decode 分支存在独立未验证路径，一并补上 _is_hip 检查；sogalin 认可 " having is_hip to protect now"。
- PR CI 无法验证改动代码 (testing): 合入前需在 AMD 上手动跑 GLM5/DeepSeek-V3.2 decode eval（clintg6 已在 MI355X 完成 TP8 实测）；PR 最终以 CI green + 人工验证合入。
- EAGLE draft runner 复用 capture() 的兼容修复 (correctness): 用 getattr(..., False) 默认值保护，且 dsa_variant 为 None 时不传额外参数调用 capture_one_shape，保持子类签名兼容。
- 环境变量应走 envs 注册表而非 os.environ.get (style): 改为 envs.SGLANG_* .get() 走统一布尔解析；该 env 后续随 clintg6 的清理要求一并删除。
- 空 batch 下 seq_lens.max() RuntimeError 防护 (correctness): 在 extend 与 eager-decode 两处 gate 前都加 numel() > 0 检查，commit 27bd826 修复。

## 风险与影响

- 风险：
  1. 双图捕获成本显性增加：每个 bs 桶多捕获一张 dense 图，代码注释自述约增加 52 张图、2 倍捕获时间，显存常驻占用随之上升，对显存紧张的部署有压力。
  2. CUDA 路径依赖 gating 正确性：共享代码靠 is_hip() 门控，若未来有人在 __init__ 或 _should_skip_logits_computation 中漏掉 _is_hip 检查，会静默改变 CUDA 上 DeepSeek-V3.2 等 DSA 模型的 decode 行为；提交历史中已发生过一次 eager 分支未同步 gating 的遗漏。

3. 子类兼容脆弱：EAGLEDraftCudaGraphRunner 复用 capture() 但不执行本类 __init__，依赖 getattr 默认值与 dsa_variant is None 的特殊调用路径，未来新增继承 capture() 的 runner 存在同类隐患。
4. 测试覆盖严重不足：无任何 PR CI 测试触及改动代码（nightly-only），回归只能靠手动 / 夜间套件；_resolve_dsa_variant 的边界（seq_lens_cpu 为空、非 contiguous、长请求混批）缺少自动化验证。
5. 混合长度性能悬崖：批内只要有一个请求 kv_len > index_topk 整批退回 sparse 图，长请求与短请求混批时收益归零。 - 影响：用户侧：AMD MI355X 上运行 GLM-5.2（及 DeepSeek-V3.2 等 DSA 模型）、输入输出长度不超过 index_topk（2048）的 decode 部署将无感获得约 2%–5% 吞吐提升与 2%–5% TPOT 下降，无需任何配置；非 DSA 模型、非 AMD 平台、长上下文（>2048）场景零影响。系统侧：CUDA 图缓存键新增 dsa_variant 维度，捕获流程从单图变为双图线性扩展，为后续按运行时属性多图捕获 + host 分发（如按长度、按稀疏度）提供了可复用的模式。团队侧：该 PR 确立了 " capture-time 分支冻结问题通过多图变体 + 变体信号解决 " 的设计范式，与既有 LoRA 双图捕获机制在 capture_mode.py 中统一。 - 风险标记：仅 AMD 验证，PR CI 无覆盖，双图显存开销，共享代码路径，无测试文件

## 关联脉络

- PR #34982 [misc] Rename shared-read boundary to shared-read ends and fix wrapper delegation: 同样修改 decode_cuda_graph_runner.py 的共享读栅栏与委托逻辑，与本 PR 同属解码 CUDA 图 runner 的机制演进，改动区域相邻。
- PR #35057 [Spec] Point multi-layer eagle's last shared-read runner at the draft runner: EAGLE draft runner 归属修复，与本 PR 中 EAGLEDraftCudaGraphRunner 复用 capture() 的兼容处理形成印证，显示 runner 子类复用链的脆弱点。
- PR #35000 Support unified SWA page mapping in attention metadata: DSA/SWA 注意力子系统的 metadata 页表映射扩展，与本 PR 的 DSA indexer 同属 DSA 注意力链路的持续演进，未来 dual-graph 需与 SWA page mapping 兼容。