PR #31477 完整报告

sgl-project/sglang

[Spec][PD] Enable fused TopK for GLM-5.2 MTP IndexShare

合并时间：2026-08-06 05:17

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31477

执行摘要

- 一句话：PD 下启用 fused DSA TopK，TPOT 降约 3.3%
- 推荐动作：值得精读。核心设计是“wire 表示不变、消费端一次性重映射”，这是跨 allocator 数据交换的经典模式，且通过 should_remap_pd_dsa_seed_to_local_slots 一个符号集中了全部可扩展门控。重点看：五重门控的条件选择、invalid_rows 的 fail-closed 语义（哨兵、越界、未分配 slot 三类校验）、以及 CI 上 e2e 多次 flaky 的处理方式；后续可关注 ROCm 侧提交的 fused TopK 扩展 PR。

功能与动机

GLM-5.2 MTP 在 draft-decode 各步之间复用 DSA IndexShare 的 top-k seed，避免重复 indexer 计算。PR #30839 引入的 seed 传输路径为保持 Prefill/Decode allocator 独立，在 wire 上传输 request-relative 位置；但现有 fused TopK 消费 allocator-local 物理 KV slot，导致 draft decode 期间回退到 unfused page-table transform，每步重复页表转换。本 PR 的目标是在 PD disaggregation 下针对 GLM-5.2 MTP IndexShare 定点优化：在 Decode 消费端一次性转换 seed 后复用既有 fused TopK，wire 表示保持不变（Prefill 侧无需改动）。

实现拆解

1. 门控决策函数：在 python/sglang/srt/layers/attention/dsa/utils.py 新增 should_remap_pd_dsa_seed_to_local_slots，用五重条件把新路径限定在极窄范围——is_cuda()、SGLANG_DSA_FUSE_TOPK=1、disaggregation_mode == "decode"、非 HiSparse、dcp_size == 1。is_cuda() 门控是 review 中 kpham-sgl 明确要求的，因为 NPU 路径（deepseek_v2_attention_mla_npu.py、eagle_draft_npu_graph_runner.py）实现不同，不能套用本次 remap。
2. fused TopK 选择逻辑调整：同文件的 should_use_dsa_fused_topk 原逻辑在 pd_index_share_seed 为真时一律禁用 fused（并留有 TODO）；现改为 not pd_index_share_seed or should_remap_pd_dsa_seed_to_local_slots(...)，即满足 remap 条件时 PD IndexShare 也可走 fused。同时按 kpham-sgl 意见补充了各 worker 角色的执行矩阵 docstring：Prefill 侧 target prefill 开、draft extend 关；Decode 侧 draft decode / target verify / draft extend 全开。
3. Decode 本地重映射：在 python/sglang/srt/speculative/eagle_disaggregation.py 的 build_eagle_disagg_draft_input 中，dsa_topk_indices 组装后、进入 EAGLE draft 循环/CUDA Graph 之前，若命中门控条件，则通过 batch.req_to_token_pool.req_to_token 按

req_pool_indices gather 出本地物理 slot；同时做整行合法性校验——位置小于 -1、超过 seq_lens、超过页表宽度、或指向未分配 slot（slot <= 0，0 是保留 padding sink）都会把整行置 -1，随后 torch.all(row < 0) 时整体回退为 None（unfused 路径），实现 fail-closed 而非消费脏数据。

4. 测试配套：test/registered/unit/disaggregation/test_disaggregation_wire.py 新增 CPU 单测 test_pd_decode_fused_topk_remaps_wire_positions_to_local_slots，覆盖 fused 路径选择与位置到本地 slot 的映射：wire 位置 [2,0,-1]、[1,3,-1] 分别映射为本地 slot [309,101,-1]、[801,990,-1]；并为既有 test_decode_input_requires_valid_seed_for_every_request 补 disaggregation_mode="null" 字段保持原语义。端到端正确性由 test_dsa_glm52_cache_layer_split.py（8-gpu-b200）覆盖，PR 期间多次 rerun 后通过。

关键文件：

- python/sglang/srt/layers/attention/dsa/utils.py（模块 稀疏注意力；类别 source；类型 core-logic；符号 should_remap_pd_dsa_seed_to_local_slots, should_use_dsa_fused_topk）：新增 should_remap_pd_dsa_seed_to_local_slots 五重门控，并改造 should_use_dsa_fused_topk 使 PD IndexShare 在可 remap 时重新启用 fused TopK，是本 PR 的生效范围控制核心。
- python/sglang/srt/speculative/eagle_disaggregation.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 build_eagle_disagg_draft_input）：build_eagle_disagg_draft_input 中加入 Decode 本地重映射逻辑，把 wire 上的 request-relative seed 转换为物理 KV slot，是本 PR 的实际生效点。
- test/registered/unit/disaggregation/test_disaggregation_wire.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_pd_decode_fused_topk_remaps_wire_positions_to_local_slots）：新增 CPU 单测覆盖 fused 路径选择与位置到本地 slot 的重映射，是本次行为唯一的确定性回归保护。

关键符号：should_remap_pd_dsa_seed_to_local_slots, should_use_dsa_fused_topk, build_eagle_disagg_draft_input

关键源码片段

python/sglang/srt/speculative/eagle_disaggregation.py

build_eagle_disagg_draft_input 中加入 Decode 本地重映射逻辑，把 wire 上的 request-relative seed 转换为物理 KV slot，是本 PR 的实际生效点。

```
# 摘自 python/sglang/srt/speculative/eagle_disaggregation.py 的 build_eagle_disagg_draft_input
    dsa_topk_indices = None
    dsa_indices_list = [req.output_dsa_topk_indices for req in batch.reqs]
    if dsa_indices_list and all(t is not None for t in dsa_indices_list):
        dsa_topk_indices = torch.stack(dsa_indices_list, dim=0).to(batch.device)
        if should_remap_pd_dsa_seed_to_local_slots(server_args):
            # wire 上传输的是 request-relative 位置（Prefill/Decode allocator 独立），
            # 而 fused TopK 消费 Decode 本地物理 KV slot，进入 draft 循环前统一 remap 一次
            req_to_token = batch.req_to_token_pool.req_to_token
            table_width = req_to_token.shape[1]
            valid_positions = dsa_topk_indices >= 0
```

```
            # clamp 只用于安全 gather，真正越界的行会在下面整行置 -1
            gather_positions = dsa_topk_indices.clamp(min=0, max=table_width - 1).to(
                torch.int64
            )
            local_slots = req_to_token[
                batch.req_pool_indices[:, None], gather_positions
            ]
            # 任一行存在非法输入即整行失效，fail-closed，避免 stale/ 越界 slot 进入 fused TopK
            invalid_rows = torch.any(
                (dsa_topk_indices < -1) # 只有 -1 是合法哨兵
                | (dsa_topk_indices >= batch.seq_lens[:, None]) # 超出请求自身长度
                | (dsa_topk_indices >= table_width) # 超出页表宽度
                # slot 0 是保留 padding sink，真实 KV 分配从 1 开始，
                # 未被触碰的 req_to_token 条目保持为 0，因此 local_slots <= 0 视为未分配
                | (valid_positions & (local_slots <= 0)),
                dim=1,
            )
            local_slots.masked_fill_(~valid_positions, -1)
            local_slots.masked_fill_(invalid_rows[:, None], -1)
            dsa_topk_indices = local_slots
        if torch.any(torch.all(dsa_topk_indices < 0, dim=1)).item():
            dsa_topk_indices = None # 整批无有效 seed，回退 unfused 路径
```

test/registered/unit/disaggregation/test_disaggregation_wire.py

新增 CPU 单测覆盖 fused 路径选择与位置到本地 slot 的重映射，是本次行为唯一的确定性回归保护。

```
# 摘自 test/registered/unit/disaggregation/test_disaggregation_wire.py
    def test_pd_decode_fused_topk_remaps_wire_positions_to_local_slots(self):
        # Prefill 侧沿 PD wire 传出的 request-relative 位置（-1 表示 padding）
        wire_positions = (
            torch.tensor([2, 0, -1], dtype=torch.int32),
            torch.tensor([1, 3, -1], dtype=torch.int32),
        )
        # Decode 本地 req_to_token 表：第 0 行保留给 padding sink，
        # 第 1/3 行分别对应两个请求的物理 KV slot 映射
        req_to_token = torch.tensor(
            [
                [0, 0, 0, 0],
                [700, 801, 902, 990],
                [410, 420, 430, 440],
                [101, 205, 309, 450],
            ],
            dtype=torch.int32,
        )
        batch = SimpleNamespace(
            reqs=[self._make_req(seed) for seed in wire_positions],
            device="cpu",
            enable_overlap=False,
```

```
        req_pool_indices=torch.tensor([3, 1], dtype=torch.int64),
        req_to_token_pool=SimpleNamespace(req_to_token=req_to_token),
        seq_lens=torch.tensor([4, 4], dtype=torch.int32),
    )
    server_args = SimpleNamespace(
        speculative_eagle_topk=1,
        speculative_num_steps=5,
        enable_multi_layer_eagle=False,
        disaggregation_mode="decode",
        enable_hisparse=False,
        dcp_size=1,
    )
    # 覆盖 SGLANG_DSA_FUSE_TOPK 与 is_cuda()，激活 Decode 本地 remap 分支
    with envs.SGLANG_DSA_FUSE_TOPK.override(True), patch(
        "sglang.srt.layers.attention.dsa.utils.is_cuda", return_value=True
    ):
        self.assertTrue(
            should_use_dsa_fused_topk(
                server_args, seed_dsa_topk_from_draft_extend=True
            )
        )
        draft_input = build_eagle_disagg_draft_input(
            batch, server_args, torch.tensor([11, 12], dtype=torch.int64), None
        )
    # 请求 0：位置 2/0 在 req_pool 行 3 中映射为 309/101，-1 保持 padding；
    # 请求 1：位置 1/3 在 req_pool 行 1 中映射为 801/990
    self.assertEqual(
        draft_input.dsa_topk_indices.tolist(),
        [[309, 101, -1], [801, 990, -1]],
    )
```

评论区精华

1. 平台门控（kpham-sgl）：指出 NPU 路径实现不同，要求 remap 门控加 is_cuda()，作者已修复；后续 tianxiaojiang4 在 gfx950 实测 fused 23.2 ms vs unfused 59.2 ms（2.55x），但门控使 ROCm 不生效，kpham-sgl 回应 "ROCm paths are untested. Can you submit a PR with tested results?"，把扩展机会留给后续 PR。
2. 行为矩阵文档化（kpham-sgl）：fused TopK 开关行为变复杂，要求记录 Prefill/Decode 各阶段执行矩阵，已写入 should_use_dsa_fused_topk 的 docstring。
3. 测试策略开放问题（kpham-sgl）：“how do we write better tests (outside of test_dsa_glm52_cache_layer_split.py)”——作者认为需要 2/4 GPU 小模型；kpham-sgl 最终 approve 时表示 "we can revisit the testing question later"，未闭环。
4. 影响范围确认（zRzRzRzRzRzR）：确认只改 PD Decode + fused TopK + 非 HiSparse + dcp_size == 1 路径后 approve。
5. CI 稳定性：test_dsa_glm52_cache_layer_split.py 在 8-gpu-b200 上连续多次失败后通过，暴露 DSA 相关 e2e 的 flaky 问题。

- is_cuda() 门控以保护 NPU 路径 (correctness): 门控函数最终以 is_cuda() 为首个条件，NPU 路径不受影响；ROCm 同样被排除（后经 tianxiaojiang4 实测确认存在 2.55× 提升空间，待后续 PR）。
- fused TopK 执行矩阵文档化 (documentation): docstring 已记录：Prefill 侧 target prefill 开、draft extend 关；Decode 侧 draft decode / target verify / draft extend 全开。
- 如何在 e2e 之外写更好的测试 (testing): kpham-sgl approve 时表示 "we can revisit the testing question later"，未闭环，留作后续工作。
- 影响范围确认 (question): 作者确认并贴出门控函数源码；随后 zRzRzRzRzRzR approve。
- ROCm 上 fused TopK 的 2.55× 扩展机会 (performance): kpham-sgl 回应 'ROCm paths are untested. Can you submit a PR with tested results?'——需要带测试结果的独立 PR 处理。

风险与影响

- 风险：
  1. decode 热路径变更：remap 位于 decode 每步的投机解码入口，但只在 graph 外执行一次、draft 循环内每步复用；128K 长上下文 A/B 实测 TPOT 下降，暂无回退风险。
  2. fail-closed 静默回退：任一行非法即整行置 -1，若整批非法会把 dsa_topk_indices 置 None 回退 unfused——不会报错，但用户可能只观察到性能下降而无显式告警，长尾场景需要可观测性支撑。
  3. 平台覆盖：is_cuda() 门控保护了 NPU 与未验证的 ROCm 路径，但 ROCm 实测 2.55× 收益暂时无法释放，属于已知未兑现的性能空间。
  4. 测试覆盖：新增仅为 CPU 单测；端到端依赖 8-gpu-b200 e2e，其历史多次失败表明回归保护不够稳定。
  5. CUDA Graph 兼容：remap 在 graph 捕获前完成，PR body 声明 target verify / draft extend / draft decode 均保持在 Full CUDA Graph 内，无 graph 重捕获风险。
- 影响：
  1. 用户侧：GLM-5.2 MTP + PD 部署下 TPOT 11.60→11.22 ms（-3.28%）、ITL -3.08%、输出吞吐 +1.11%，MTP acceptance 与每迭代解码 token 数不变（66%、2.96），131072+1536 长上下文下收益稳定。
  2. 系统侧：每 decode 迭代消除 2 个 page-table transform kernel（prefill/decode 变体）并移除 704 次 unfused topk_kernel 调用，转为 fused topk_small_batch_kernel<true>；indexer scoring/planning 工作完整保留。
  3. 团队侧：补上了 PR #30839 在 should_use_dsa_fused_topk 里留下的 TODO，kernel 选择、wire 传输、消费端 remap 三者协同设计路径清晰，为 ROCm/NPU 扩展留下明确门控点。 - 风险标记：decode 热路径变更，CUDA-only 门控（ROCm 收益未释放），非法 seed 静默回退 unfused，e2e 测试多次 flaky，新增覆盖仅 CPU 单测

关联脉络

- PR #30839 DSA IndexShare seed-transfer path（PR body 引用）：本 PR 的直接前置：其引入的 request-relative seed 传输路径导致 Decode 侧 fused TopK 回退 unfused；本

PR 在 Decode 消费端重映射后重新启用 fused TopK，且旧代码中 kpham-sgl 留的 TODO 正是本 PR 的动机来源。