PR #26239 完整报告

sgl-project/sglang

[dsv4] fix multi-step draft on non-cuda-graph path

合并时间：2026-05-25 08:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26239

执行摘要

- 一句话：修复 DSv4 非 cuda-graph 路径下 multi-step draft 的 KV 写入布局错误
- 推荐动作：建议合并。该修复针对明确 bug，方案简洁且提取了共享逻辑，有助于后期维护。后续可考虑增加测试覆盖非 cuda-graph 的 draft 路径。

功能与动机

DSv4 后端在 init_forward_metadata 时间将每步的 KV 写目标（c4_out_loc/c128_out_loc）烘焙到注意力元数据中——不同于 FlashMLA / FlashInfer / Triton 仅在 forward_* 中消费 out_cache_loc。默认的 cuda-graph 捕获 / 重放路径绕过了 init_forward_metadata，因此该 bug 仅在 --disable-cuda-graph 时触发。PR #23882 引入了该路径，但 CI 未覆盖（来自 PR body）。

实现拆解

1. 在 eagle_utils.py 中新增 per_step_draft_out_cache_loc 函数：作为多步 draft 的 out_cache_loc 布局的唯一权威来源，将形状从 [bs * topk * num_steps] 重塑为 [num_steps, bs * topk] 视图，并包含形状断言。
2. 修改 eagle_worker_v2.py 的 draft_forward 方法：将原有的内联 reshape + permute 替换为调用新辅助函数，消除重复实现并确保一致性。
3. 修改 deepseek_v4_backend.py 的 init_forward_metadata 方法：在 decode 分支中新增判断，当 topk > 0 and speculative_num_steps > 1 时，调用辅助函数获取当前步的切片传递给 init_forward_metadata_decode，解决了断言形状不匹配的问题。同时添加了导入语句。

关键文件：

- python/sglang/srt/speculative/eagle_utils.py（模块 推测解码；类别 source；类型 core-logic；符号 per_step_draft_out_cache_loc）：新增核心辅助函数 per_step_draft_out_cache_loc，定义了多步 draft 的 out_cache_loc 布局，被另外两个文件引用。
- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力层；类别 source；类型 dependency-wiring）：在 init_forward_metadata 的 decode 分支中增加了 multi-step draft 时的切片逻辑，是修复断言失败的关键。

- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 dependency-wiring）：在 draft_forward 中将原有的内联 reshape 替换为调用共享辅助函数，消除代码重复。

关键符号：per_step_draft_out_cache_loc, DeepseekV4AttnBackend.init_forward_metadata, EagleWorkerV2.draft_forward

关键源码片段

python/sglang/srt/speculative/eagle_utils.py

新增核心辅助函数 per_step_draft_out_cache_loc，定义了多步 draft 的 out_cache_loc 布局，被另外两个文件引用。

```
def per_step_draft_out_cache_loc(
    out_cache_loc: torch.Tensor,
    batch_size: int,
    topk: int,
    num_steps: int,
) -> torch.Tensor:
    """从多步 EAGLE draft 的 out_cache_loc 缓冲区中提取 per-step 切片。

    作为 EagleWorkerV2.draft_forward (per-step 写目标) 和 DeepseekV4AttnBackend
    (per-step 压缩写目标，烘焙到 metadata 中) 共享布局的唯一权威来源。
    """
    expected = batch_size * topk * num_steps
    assert out_cache_loc.shape[0] == expected, (
        f"out_cache_loc.shape[0]={out_cache_loc.shape[0]} != "
        f"batch_size * topk * num_steps = {batch_size}*{topk}*{num_steps}={expected}"
    )
    # 视图 [bs, topk, num_steps] -> permute [num_steps, bs, topk] -> reshape [num_steps,
    bs*topk]
    # 这样 out_cache_loc[i] 就是第 i 步所有 batch 和 topk 位置的写目标
    return (
        out_cache_loc.view(batch_size, topk, num_steps)
        .permute(2, 0, 1)
        .reshape(num_steps, -1)
    )
```

python/sglang/srt/layers/attention/deepseek_v4_backend.py

在 init_forward_metadata 的 decode 分支中增加了 multi-step draft 时的切片逻辑，是修复断言失败的关键。

```
if forward_batch.forward_mode.is_decode_or_idle():
    # DSv4 将当前步的 KV 写目标 (c4/c128) 烘焙到 metadata 中，
    # 所以此时就要对共享的多步 out_cache_loc 进行切片，而不是在 forward 时再做。
    out_cache_loc = forward_batch.out_cache_loc
    if self.topk > 0 and self.speculative_num_steps > 1:
        # 在 multi-step draft 时，out_cache_loc 是 [bs*topk*num_steps] 的扁平张量
        # 这里取出当前步 (self.speculative_step_id) 对应 [bs*topk] 的部分
```

```
            out_cache_loc = per_step_draft_out_cache_loc(
                out_cache_loc,
                forward_batch.batch_size,
                self.topk,
                self.speculative_num_steps,
            )[self.speculative_step_id]
        metadata = self.init_forward_metadata_decode(
            max_seq_len=max_seq_len,
            req_pool_indices=req_pool_indices,
            seq_lens=seq_lens,
            out_cache_loc=out_cache_loc,
        )
```

python/sglang/srt/speculative/eagle_worker_v2.py

在 draft_forward 中将原有的内联 reshape 替换为调用共享辅助函数，消除代码重复。

```
def draft_forward(self, forward_batch: ForwardBatch):
    # ... 其他代码 ...
    out_cache_loc = forward_batch.out_cache_loc
    # 使用共享辅助函数替代原有的内联 reshape + permute
    out_cache_loc = per_step_draft_out_cache_loc(
        out_cache_loc,
        forward_batch.batch_size,
        self.topk,
        self.speculative_num_steps,
    )
    # 后续循环中通过 out_cache_loc[i] 获取第 i 步的切片
    for i in range(self.speculative_num_steps):
        # ... 省略 ...
        forward_batch.out_cache_loc = out_cache_loc[i]
        # ... 省略 ...
```

评论区精华

无 review 评论，PR 由作者直接合并。PR body 清晰说明了 bug 背景、触发条件和修复方案。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更集中在非 cuda-graph 路径，cuda-graph 路径不受影响。新增的辅助函数与原有逻辑等价（代码层面是提取和共享），且添加了形状断言。但缺少直接针对该路径的单元测试，回归风险依赖于集成测试。
- 影响：影响范围：仅限 DeepSeek V4 模型且使用 --disable-cuda-graph 的 EAGLE multi-step draft 场景，修复后该场景可用。对 cuda-graph 路径和其他模型无影响。
- 风险标记：缺少测试覆盖，核心路径变更

关联脉络

- PR #23882 引入非 cuda-graph 的 multi-step draft 路径：本 PR 修复了 PR #23882 引入的路径中的 bug。