PR #27338 完整报告

sgl-project/sglang

[Bug] Fix EAGLE draft CUDA-graph `kv_indices` under-allocation for `topk > 1`

合并时间：2026-06-06 04:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27338

执行摘要

- 一句话：修复 EAGLE 推测解码 topk>1 时 CUDA Graph 内存越界
- 推荐动作：此 PR 值得合并，修复了明确的严重 bug。对于从事推测解码或 CUDA Graph 相关开发的工程师，建议精读 common_template 中的断言实现，它提供了一种低成本、高覆盖的防御性编程模式。

功能与动机

当 speculative_eagle_topk > 1 且并发请求数较高时，generate_draft_decode_kv_indices 内核写入超出 CUDA Graph 分配的 kv_indices 缓冲区边界，导致静默内存损坏（错误的 draft token）或偶发崩溃（vectorized_gather_kernel 索引越界）。该问题由 Issue #26340 跟踪。PR body 指出 init_cuda_graph_state 中缓冲区缺少 topk 因子，而 eager 路径的 init_forward_metadata 已正确包含该因子。

实现拆解

修复在单个文件 python/sglang/srt/layers/attention/flashinfer_backend.py 中完成：

1. 修正缓冲区大小：在 EagleDraftFlashInferAttnBackend.init_cuda_graph_state 中，将 cuda_graph_kv_indices 张量的第二维从 max_bs * max_context_len 改为 max_bs * self.topk * max_context_len，与 eager 路径中的 init_forward_metadata 分配保持一致。
2. 添加运行时断言：在 common_template 方法中（所有路径共享的入口），增加了 required_kv_indices_len 的计算和断言，检查当前批次需要的条目数是否不超过缓冲区列宽。如果越界，则抛出带有详细信息的 AssertionError，将静默溢出转化为确定性失败。
3. 原逻辑无需修改：common_template 中 kv_indices_buffer[i] 的切片 [: seq_lens_sum * self.topk + bs * (i + 1)] 已经正确考虑了 topk 因子，仅缓冲区分配错误。
4. 变更精简：经过多个 commit 迭代后，移除了临时添加的溢出探测工具和专用测试类，最终只保留了核心修复和断言。

关键文件：

- python/sglang/srt/layers/attention/flashinfer_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 init_cuda_graph_state, common_template）：包含所有变更：缓冲区大小修正和运行时断言添加。

关键符号：common_template, init_cuda_graph_state

关键源码片段

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

包含所有变更：缓冲区大小修正和运行时断言添加。

```
# python/sglang/srt/layers/attention/flashinfer_backend.py

class EagleDraftFlashInferAttnBackend:

    def common_template(
        self,
        forward_batch: ForwardBatch,
        kv_indices_buffer: torch.Tensor,
        call_fn: Callable,
    ):
        num_seqs = forward_batch.batch_size
        bs = self.topk * num_seqs
        seq_lens_sum = forward_batch.seq_lens_sum

        # 运行时断言：检查缓冲区是否足够容纳当前批次的数据
        # generate_draft_decode_kv_indices 内核会将 topk 个分支序列打包到每一行
        # 需要的总条目数为：seq_lens_sum * topk（每个序列的 KV 索引）
        # + bs * num_steps（每步每个分支一个额外索引）
        required_kv_indices_len = (
            seq_lens_sum * self.topk + bs * self.speculative_num_steps
        )
        assert required_kv_indices_len <= kv_indices_buffer.shape[1], (
            f"EAGLE draft kv_indices row too small: need {required_kv_indices_len} "
            f"but row width is {kv_indices_buffer.shape[1]} (topk={self.topk}, "
            f"num_seqs={num_seqs}, seq_lens_sum={seq_lens_sum}, "
            f"num_steps={self.speculative_num_steps}); the buffer must be sized "
            f"max_bs * topk * max_context_len."
        )

        # ... 后续 kernel launch 和调用逻辑 ...

    def init_cuda_graph_state(self, max_bs: int, max_num_tokens: int):
        # 修复：将第二维从 max_bs * max_context_len 改为 max_bs * self.topk * max_context_len
        # 因为 generate_draft_decode_kv_indices 会为每个序列打包 topk 个分支
        self.cuda_graph_kv_indices = torch.zeros(
            (self.speculative_num_steps, max_bs * self.topk * self.max_context_len),
            dtype=torch.int32,
            device="cuda",
        )
        # ... 后续初始化 ...
```

评论区精华

该 PR 的 review 评论数为 0，讨论主要在 PR body 中由作者 hnyls2002 完成。作者详细描述了问题的根源（init_cuda_graph_state 缺少 topk 因子）、表现（静默内存损坏或崩溃）、以及修复方法。PR 还包含了隔离问题的复现脚本和确定性断言示例。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：低。缓冲区大小从 max_bs * max_context_len 增大到 max_bs * topk * max_context_len，只会增加内存占用而不会影响现有逻辑。断言是新增检查，不会改变已有行为。
  2. 性能风险：低。缓冲区增大可能导致 CUDA Graph 内存占用小幅增加（取决于 topk 值），但通常 topk 为 2-8，内存增量可控。断言仅在 host 端执行一次，开销可忽略。
  3. 兼容性风险：无。仅影响 topk > 1 的 EAGLE 推测解码路径，该功能相对较新，不影响其他模式。
- 影响：
  1. 用户影响：修复了 speculative_eagle_topk > 1 时 CUDA Graph 模式下的崩溃和静默错误，提高了模型推理的稳定性和正确性。受影响的用户将不再遇到偶发的 out-of-vocab 错误或错误的 draft token。
  2. 系统影响：仅修改了一个文件中的两处代码，变更范围极小。
  3. 团队影响：无。 - 风险标记：核心路径变更，内存分配修正

关联脉络

- PR #26340 flashinfer topk>1 draft coredump: 此 PR 直接修复了该 issue 中报告的崩溃问题。