PR #27360 完整报告

sgl-project/sglang

[Spec] Fix fa3 EAGLE draft-decode expand page_table scatter OOB for topk>1 + page_size>1

合并时间：2026-06-06 15:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27360

执行摘要

- 一句话：修复 fa3 EAGLE draft-decode page_table scatter OOB
- 推荐动作：值得合并与精读。本 PR 修复了一个隐蔽的静默内存损坏 bug，展示了 cuda-graph 元数据构造中一个微妙的维度不匹配问题。建议关注： 1) cache_loc 切片与 page_size == 1 分支的对齐设计； 2) 始终启用断言作为安全网的做法； 3) revert 开关的注册方式，这是一种低成本 A/B 调试基础设施。

功能与动机

PR body 明确指出：FlashAttentionMultiStepBackend 构建 EAGLE draft-decode 展开元数据时，page_size > 1 分支未将 cache_loc 切片到 decode_length，而 page_size == 1 分支已有正确切片。对于 topk > 1，每个分支的 draft slot 跨越的页数超出 page_table 行容量，scatter_ 越界写入，静默损坏 cuda-graph 池，最终表现为非法内存访问或 NaN logits。

实现拆解

1. 切片修复：在 FlashAttentionBackend._apply_cuda_graph_metadata 中，page_size > 1 分支调用 draft_decode_set_expand_metadata 之前，加入 cache_loc = cache_loc[:, :decode_length]，将 num_steps 宽的 cache_loc 截取为当前步数有效的 decode_length，与 page_size == 1 分支的行为一致。
2. 始终启用的尺寸断言：在上述切片之后，添加 assert cache_loc.shape[1] <= metadata_expand.page_table.shape[1]，若未来回归导致越界，会在出问题时立即失败，而非静默损坏内存。该断言无环境标志，不触发 torch.compile 图断裂。
3. 注册 revert 开关：在 pr_fix_toggle.py 中为 PR #27360 注册 revert YAML 补丁，允许通过环境变量 SGLANG_DEBUG_REVERT_PR=27360 反向应用修复，便于 A/B 调试。
4. 安全处理空批次：次提交（3378283）将 draft_decode_set_expand_metadata 中的 positions.max() 前使用 torch.nn.functional.pad 填充 -1 哨兵，确保空批次（num_seqs == 0）下不会因空张量 max() 抛出运行时错误，且保持 torch.compile 兼容。

关键文件：

- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _apply_cuda_graph_metadata, draft_decode_set_expand_metadata）：核心修复文件，修改 _apply_cuda_graph_metadata 方法中的 page_size > 1 分支，添加 cache_loc 切片和始

终启用的尺寸断言，防止 page_table scatter 越界。

- python/sglang/srt/debug_utils/pr_fix_toggle.py（模块 调试工具；类别 source；类型 core-logic；符号 _PR_REVERT_YAML_27360, _PR_FIX_REVERT_YAML）：注册 PR #27360 的 revert 补丁，允许通过环境变量 SGLANG_DEBUG_REVERT_PR=27360 反向应用修复，用于 A/B 调试。

关键符号：FlashAttentionBackend._apply_cuda_graph_metadata, draft_decode_set_expand_metadata

关键源码片段

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

核心修复文件，修改 _apply_cuda_graph_metadata 方法中的 page_size > 1 分支，添加 cache_loc 切片和始终启用的尺寸断言，防止 page_table scatter 越界。

```
# File: python/sglang/srt/layers/attention/flashattention_backend.py
# Function: FlashAttentionBackend._apply_cuda_graph_metadata (partial)

if self.page_size > 1:
    # Only the draft tokens produced up to this step are live;
    # cache_loc arrives num_steps-wide. Slice so the scatter fills at
    # most decode_length of the (decode_length + 1) expand page_table
    # columns -- without this the extra distinct pages overflow the row.
    cache_loc = cache_loc[:, :decode_length]
    assert (
        cache_loc.shape[1] <= metadata_expand.page_table.shape[1]
    ), (
        f"draft expand page_table too narrow: cache_loc width "
        f"{cache_loc.shape[1]} > "
        f"{metadata_expand.page_table.shape[1]} columns "
        f"(decode_length + 1); page_size={self.page_size}, "
        f"topk={self.topk}, num_steps={self.speculative_num_steps}"
    )
    draft_decode_set_expand_metadata(
        cache_seqlens_int32=metadata_expand.cache_seqlens_int32,
        page_table=metadata_expand.page_table,
        last_page_lens=last_page_lens,
        decode_length=decode_length,
        cache_loc=cache_loc,
        topk=self.topk,
        page_size=self.page_size,
    )
else:
    num_seqs = cache_loc.shape[0]
    metadata_expand.page_table[:num_seqs, :decode_length].copy_(
        cache_loc[:, :decode_length]
    )

# File: python/sglang/srt/layers/attention/flashattention_backend.py
```

```
# Function: draft_decode_set_expand_metadata (partial)

# Note: cache_loc is pre-sliced to decode_length by the caller, so the scatter fills
# at most decode_length of the (decode_length + 1) page_table columns.
# Vectorized torch.unique_consecutive: track value change points then scatter
mask = torch.ones_like(cache_loc, dtype=torch.bool)
mask[:, 1:] = cache_loc[:, 1:] != cache_loc[:, :-1]
positions = mask.cumsum(dim=1) - 1
num_seqs = cache_loc.shape[0]
# Safeguard against empty batch: pad with a sentinel -1 so that .max() on
# an empty tensor doesn't raise RuntimeError under torch.compile.
if num_seqs == 0:
    num_seqs_padded = 1
    positions = torch.nn.functional.pad(positions, (0, 0, 0, 1), value=-1)
else:
    num_seqs_padded = num_seqs
max_positions = positions.max().item() + 1
...
```

评论区精华

Review 中 gemini-code-assist[bot] 指出：若 num_seqs == 0（如空批次或 warmup 场景），positions 可能是空张量，调用 .max() 会触发 RuntimeError: zero-dimensional tensor cannot be reduced，建议使用 torch.nn.functional.pad 填充安全默认值（如 -1）后再求最大值，以保持 torch.compile 下无图断裂。hnyls2002 回应已在 33782830dd 提交中处理。此问题不影响最终合并。

- 空批次下 positions.max() 可能崩溃 (correctness): hnyls2002 在 3378283 中通过 pad 填充 -1 哨兵修复，确保空批次安全且不触发 torch.compile 图断裂。

风险与影响

- 风险：
  1. 回归风险低：修复仅在 page_size > 1 分支添加切片和断言，不影响其他路径。assert 仅在越界时触发，不会改变正常行为的计算结果。
  2. 性能影响：切片是视图操作，无额外内存拷贝；assert 仅在 cuda-graph 元数据设置阶段执行一次，非热点路径，对推理延迟无影响。
  3. 空批次安全：已通过 pad 空张量处理确保 assert 不崩溃。
  4. 调试便利：revert 开关允许无需代码修改即可回退修复，方便比对外部因素引起的同类问题。 - 影响：直接修复 EAGLE draft-decode 场景（topk>1 且 page_size>1，如 EAGLE3 默认 topk=8 和 page_size=2）下的 cuda-graph 内存损坏问题，具体表现为 NaN logits 或非法内存访问。该场景在用户使用 fa3 attention backend 进行投机解码时可能遇到。修复后提供确定性断言，使问题可在早期定位。影响范围限于使用 FlashAttentionMultiStepBackend 进行 EAGLE draft-decode 的推理路径，对其他 attention backend 无影响。 - 风险标记：核心路径变更，新增 assert 可能影响编译，需要关注空批次边界条件

关联脉络

- PR #27428 [debug] Register #27338 EAGLE draft kv_indices revert in pr_fix_toggle: 同样在 pr_fix_toggle.py 中注册 revert 补丁，与本 PR 复用同一调试基础设施模式。
- PR #27114 [Bugfix] Restore overridden HF config fields and support index_skip_topk_offset for DSA topk sharing: 同为投机解码相关 bugfix，涉及 EAGLE 和 topk 共享逻辑。