PR #36658 完整报告

sgl-project/sglang

[multimodal_gen] fix: make tail_attn_meta CUDA-graph capturable

合并时间：2026-08-28 12:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36658

执行摘要

- 一句话：修复 tail_attn_meta 在 CUDA graph 捕获中的 H2D 拷贝问题
- 推荐动作：值得精读。该 PR 是一个小而精确的修复，展示了在 CUDA graph 捕获中避免 host 到 device 拷贝的最佳实践，并提供了明确的回归测试。它对于理解 SP 分片与 CUDA graph 的交互以及潜在的性能陷阱很有参考价值。

功能与动机

PR 描述中指出 tail_attn_meta 在 DiT 前向中被 breakable-cuda-graph 捕获，而 torch.tensor([valid, num_pad], device=cuda) 通过页式主机内存暂存 Python 列表，在 CUDA graph 捕获期间执行 H2D 拷贝会触发 Cannot copy between CPU and CUDA tensors during CUDA graph capture 错误，导致整个签名捕获失败并静默回退到 eager 模式。虽然当前默认配置下没有模型会触发此路径，但 Klein 模型固定 512 的 SP 大小无法整除非 2 的幂次（如 --ulysses-degree 3），且任何 SP 模型在捕获的前向中遇到尾部填充都会立即触发问题。

实现拆解

1. 修改 sp_shard_utils.tail_attn_meta：移除 row = torch.tensor([valid, num_pad], ...) 和 seglens = row.repeat(batch_size) 的行，改用 torch.arange(batch_size, device=device) * seq 生成行起始偏移，然后通过 cu_seqlens[1::2] = row_starts + valid 和 cu_seqlens[2::2] = row_starts + seq 直接填充累积段长度数组。这样完全在设备端构造张量，避免了页式主机内存的暂存。
2. 新增回归测试 test_tail_meta_is_cuda_graph_capturable：在 test_sp_shard.py 中添加该测试，使用 skipif 跳过无 CUDA 的环境，测试在 torch.cuda.graph 上下文中捕获 tail_attn_meta 的构建过程，并验证捕获结果与 eager 模式结果一致。
3. 测试验证：现有 CPU 值测试（test_tail_meta_*）继续通过，整个 test_sp_shard.py 套件共 24 个测试全部通过，确保数值不变。

关键文件：

- python/sglang/multimodal_gen/runtime/distributed/sp_shard_utils.py（模块 多模态生成；类别 source；类型 core-logic；符号 tail_attn_meta）：核心逻辑变更：修改 tail_attn_meta 函数，避免 host 到 device 的拷贝，使 cu_seqlens_tail 可在 CUDA graph 捕获期间构建。

- python/sglang/multimodal_gen/test/unit/test_sp_shard.py（模块 序列并行；类别 test；类型 test-coverage；符号 test_tail_meta_is_cuda_graph_capturable）：新增回归测试，验证 CUDA graph 捕获下的行为，并确保捕获结果与 eager 模式一致。

关键符号：tail_attn_meta, test_tail_meta_is_cuda_graph_capturable

关键源码片段

python/sglang/multimodal_gen/runtime/distributed/sp_shard_utils.py

核心逻辑变更：修改 tail_attn_meta 函数，避免 host 到 device 的拷贝，使 cu_seqlens_tail 可在 CUDA graph 捕获期间构建。

```
# python/sglang/multimodal_gen/runtime/distributed/sp_shard_utils.py
# 关键函数：tail_attn_meta
# 该函数在每个 SP 分片的前向中被调用，构建 varlen FA 的累积段长度。
# 原实现使用 torch.tensor([valid, num_pad], device=cuda) 会通过页式主机内存暂存，
# 在 CUDA graph 捕获时触发 'Cannot copy between CPU and CUDA tensors' 错误。
# 新实现全部在设备端用 arange 算术构造，避免 H2D 拷贝，使捕获能够成功。
def tail_attn_meta(
    shard: SpShard,
    batch_size: int,
    device: torch.device,
    image_seq_len: int = 0,
) -> dict | None:
    # 非 SP 或无填充时直接返回 None，走普通路径
    if shard.sp_size <= 1 or shard.num_pad == 0:
        return None
    seq = shard.sp_size * (shard.local_len + image_seq_len)
    valid = seq - shard.num_pad
    # 生成每个 batch 行的起始偏移，全部在设备端进行，避免 host-staged 拷贝
    row_starts = torch.arange(batch_size, dtype=torch.int32, device=device) * seq
    cu_seqlens = torch.zeros(2 * batch_size + 1, dtype=torch.int32, device=device)
    # 使用步长 2 分别填充 [valid] 和 [seq] 位置，等价于原来的 cumsum 逻辑
    cu_seqlens[1::2] = row_starts + valid
    cu_seqlens[2::2] = row_starts + seq
    return {
        "pad_start": valid,
        "pad_end": seq,
        "local_pad": shard.local_pad,
        "cu_seqlens_tail": cu_seqlens,
        "max_seqlen_tail": max(valid, shard.num_pad),
    }
```

python/sglang/multimodal_gen/test/unit/test_sp_shard.py

新增回归测试，验证 CUDA graph 捕获下的行为，并确保捕获结果与 eager 模式一致。

```
# python/sglang/multimodal_gen/test/unit/test_sp_shard.py
# 新增回归测试：验证 tail_attn_meta 在 CUDA graph 捕获期间可成功构造，
# 并且捕获的结果与 eager 模式下的一致。
```

```
import pytest
import torch


@pytest.mark.skipif(not torch.cuda.is_available(), reason="needs CUDA graph capture")
def test_tail_meta_is_cuda_graph_capturable():
    """该元数据在 DiT forward 中被 breakable-cuda-graph 捕获；
    如果使用 host-staged tensor 构造，会因 'Cannot copy between CPU and CUDA tensors'
    中止捕获。"""
    device = torch.device("cuda")
    shard = SpShard(orig_len=15, local_len=8, num_pad=1, sp_size=2, sp_rank=1)
    # 直接在 eager 模式获取参考结果
    eager = tail_attn_meta(shard, 2, device, image_seq_len=100)

    # 预热 side stream，模拟真实捕获前的流切换
    side = torch.cuda.Stream()
    side.wait_stream(torch.cuda.current_stream())
    with torch.cuda.stream(side):
        tail_attn_meta(shard, 2, device, image_seq_len=100)
    torch.cuda.current_stream().wait_stream(side)

    # 在 CUDA graph 捕获上下文中构造，验证不会报错
    graph = torch.cuda.CUDAGraph()
    with torch.cuda.graph(graph):
        captured = tail_attn_meta(shard, 2, device, image_seq_len=100)
    graph.replay()
    torch.cuda.synchronize()
    # 捕获结果与 eager 结果必须一致
    assert torch.equal(captured["cu_seqlens_tail"], eager["cu_seqlens_tail"])
```

评论区精华

本 PR 只有一个评论（作者 /tag-and-rerun-ci），无 reviewer 的实质讨论。Reviewer mickqian 已批准该 PR，无其他评论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 数值一致性风险：替换实现后，cu_seqlens_tail 的构造方式从 cumsum 改为直接按步长填充，如果 seq 或 valid 的计算有误，可能导致数值不一致。但现有 CPU 值测试已覆盖多种场景，且新增的 CUDA graph 测试验证了捕获结果的正确性。
  2. CUDA graph 捕获的流要求：测试中使用了 side stream 来预热捕获，这模仿了实际捕获中的流切换，但实际模型中可能存在更复杂的流切换情况，需要关注极端场景下的捕获失败。

3. 性能影响：新实现使用 *arange* 和直接索引赋值，理论上比 cumsum 更快，但由于它只在捕获路径中运行，对整体性能影响微乎其微。

- 影响：
  1. 用户影响：此修复使得 SP 分片在有尾部填充时不再静默回退到 *eager* 模式，而是能成功捕获 CUDA graph，从而保持性能。对于使用非 2 的幂次 SP 大小（如 --ulysses-degree 3）的用户尤其重要。
  2. 系统影响：修复了 BCG（Breakable CUDA Graph）与 SP 组合场景下的潜在性能退化，确保捕获路径的完整性。
  3. 团队影响：新增的回归测试作为持续保障，防止未来重新引入 host-staged tensor 构造。该变更涉及 multimodal_gen 的 sp_shard_utils，与相关模型（如 flux_2、qwen_image、mova）的 CUDA graph 捕获相关，但这些模型均未在默认配置下受影响，影响范围有限。
     - 风险标记：CUDA graph 捕获路径变更，数值一致性需回归验证

关联脉络

- PR #36705 [HiCache] Stop populating host-pool mmaps twice (-13% allocation time): 同为 memory/host 相关性能优化，涉及 host 内存操作，与本 PR 的 host-staged tensor 问题有相似主题。
- PR #36288 [1/N][Mix] Mixed Chunk Prefill Base: 涉及 DP attention 和调度，本 PR 的 tail padding 与 SP shard 路径与此相关。