PR #26442 完整报告

sgl-project/sglang

Revert "improve: combine vit calls for images from different reqs from one batch (#25910)"

合并时间：2026-05-30 02:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26442

执行摘要

- 一句话：回滚 #25910 VLM 批处理优化，修复 AMD CI 崩溃与性能回退
- 推荐动作：建议所有涉及 VLM 多模态编码的开发者精读此 PR，特别是 _get_chunked_prefill_embedding 函数中 torch.split 的使用陷阱。回滚本身是安全的，但值得关注 #25910 中暴露的设计问题：当模型编码器返回的 embedding 长度与输入侧的占位符跨度不一致时，必须通过实际返回的行数来驱动分割，而非假设 sum(end-start+1) 一致。后续重新实现批处理优化时应优先采纳这一教训。

功能与动机

PR 提交者指出，自 #25910 合并后 AMD CI 一直处于断裂状态，存在至少三种不同的 VLM 症状，但根因相同——跨请求批处理路径中 torch.split 使用的占位符 token 计数与模型实际返回的 embedding 长度不匹配。PR body 引用了 CI 运行日志和 git blame 定位到 mm_utils.py:684，并展示了回滚后吞吐从 857 token/s 恢复到 9443 token/s，准确率恢复到 0.44 的对比数据。

实现拆解

1. 纯回滚策略：基于 git revert fa6f4dfb3 生成反向变更，精确撤销 #25910 引入的所有修改，文件范围一致（mm_utils.py 和 schedule_batch.py）。
2. 冲突处理：mm_utils.py 中 #26167 新增的 _can_skip_pre_embed_feature_move 函数已触及同一区域，回滚时从冲突标记中取 pre-#25910 一侧，保留该函数体及其在旧路径中的调用；_move_items_to_device 内部的 _cpu_feature 保存逻辑被恢复为原文（去掉保存 CPU 引用的注释和赋值）。
3. 符号重命名复原：原本被 #25910 重命名的 get_chunked_embedding_legacy → _get_chunked_embedding_full，find_chunk_items_and_check_cache / assemble_chunk_embedding → _get_chunked_embedding_by_item，以及新引入的 get_chunked_prefill_embedding_legacy 和 get_multimodal_data_bounds 全部被移除，回到 #25910 之前的函数名和实现。
4. 数据成员回退：schedule_batch.py 的 MultimodalDataItem 类中删除了 _cpu_feature 字段定义（#25910 新增）。
5. 测试与配置：无测试文件变更，仅源码还原。

关键文件：

- python/sglang/srt/managers/mm_utils.py（模块 多模态；类别 source；类型 core-logic；符号 get_chunked_embedding_legacy, _get_chunked_embedding_full, find_chunk_items_and_check_cache, _get_chunked_embedding_by_item）：核心变更文件，回滚了 #25910 中引入的跨请求批处理路径，恢复了逐图像 / 逐请求编码的策略。涉及函数重命名、逻辑简化和 get_multimodal_data_bounds 工具的移除。
- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic）：伴随变更，删除了 #25910 新增的 _cpu_feature 字段，该字段用于保存 GPU 迁移前的 CPU 引用以加速 offload，回滚后不再需要。

关键符号：get_chunked_embedding_legacy (恢复为原实现), _get_chunked_embedding_full (原函数，已恢复原名), find_chunk_items_and_check_cache (恢复), assemble_chunk_embedding (恢复), _get_chunked_embedding_by_item (被移除), get_chunked_prefill_embedding_legacy (恢复), get_multimodal_data_bounds (被移除)

关键源码片段

python/sglang/srt/managers/mm_utils.py

核心变更文件，回滚了 #25910 中引入的跨请求批处理路径，恢复了逐图像 / 逐请求编码的策略。涉及函数重命名、逻辑简化和 get_multimodal_data_bounds 工具的移除。

```
# === mm_utils.py 核心变更对照 ===
# 回滚后，_get_chunked_embedding_by_item 被移除，
# 恢复原有的 find_chunk_items_and_check_cache 和 assemble_chunk_embedding 函数

# 恢复后的辅助函数：找出当前块重叠的 items，并检查缓存
def find_chunk_items_and_check_cache(
    embedding_items_per_req: List[MultimodalDataItem],
    items_offset: List[Tuple[int, int]],
    chunk_start: int,
    chunk_end: int,
) -> List[Tuple[MultimodalDataItem, Optional[torch.Tensor], int, int]]:
    """
    返回 (item, cached_embedding_or_None, start, end) 列表，
    这些 item 的区间与 [chunk_start, chunk_end) 重叠。
    """
    chunk_entries = []
    for item, (start, end) in zip(embedding_items_per_req, items_offset):
        if end >= chunk_start and start < chunk_end:
            cached = embedding_cache.get_single(item.hash)
            emb = cached.embedding if cached is not None else None
            chunk_entries.append((item, emb, start, end))
    return chunk_entries

# 恢复后的辅助函数：从缓存或新编码结果中切出重叠部分的 embedding 并拼接
def assemble_chunk_embedding(
    chunk_entries: List[Tuple[Any, torch.Tensor, int, int]],
    chunk_start: int,
```

```
    chunk_end: int,
) -> Optional[torch.Tensor]:
    """
    对每个 item，取其 embedding 中落在 [chunk_start, chunk_end) 的部分，
    最后将所有切片拼接成连续的块。
    """
    chunk_slices = []
    for _, emb, start, end in chunk_entries:
        overlap_start = max(start, chunk_start)
        overlap_end = min(end, chunk_end - 1) # inclusive
        local_start = overlap_start - start
        local_end = overlap_end - start + 1 # exclusive for slicing
        chunk_slices.append(emb[local_start:local_end])

    if not chunk_slices:
        return None
    return torch.cat(chunk_slices, dim=0)

# 恢复后的全局状态路由函数（原 get_chunked_prefill_embedding_legacy 改名回）
def get_chunked_prefill_embedding_legacy(
    data_embedding_func: DataEmbeddingFunc,
    embedding_items: List[MultimodalDataItem],
    items_size: List[int],
    prefix_length: List[int],
    extend_length: List[int],
    items_offset_list: List[List[Tuple[int, int]]],
    input_ids: torch.Tensor,
    device: torch.device,
) -> torch.Tensor:
    """
    走原有逐图像/逐请求路径：
    先按 chunk 范围过滤，然后单独编码缺失 item，
    最后用 assemble_chunk_embedding 拼接。
    不再尝试跨请求批处理 ViT 调用。
    """
    # ... 略过具体实现，与 #25910 之前相同
    pass
```

python/sglang/srt/managers/schedule_batch.py

伴随变更，删除了 #25910 新增的 _cpu_feature 字段，该字段用于保存 GPU 迁移前的 CPU 引用以加速 offload，回滚后不再需要。

```
# schedule_batch.py 中 MultimodalDataItem 类的变更
@dataclasses.dataclass
class MultimodalDataItem:
    modality: Modality
    hash: int = None
    pad_value: int = None
    offsets: Optional[list] = None
```

```
    format: MultimodalInputFormat = MultimodalInputFormat.NORMAL

    # 原始特征（processor 返回的 pixel_values 等）
    feature: Union[torch.Tensor, np.ndarray] = None
    # 回滚后：移除 _cpu_feature 字段（#25910 引入，用于保存 CPU 引用避免再拷贝）
    # 该字段只在 #25910 的批处理路径中有意义，回滚后不再使用
    # 预计算 embedding（预编码的 encoder 输出）
    precomputed_embeddings: Optional[Union[torch.Tensor, np.ndarray]] = None

    # ... 其余方法不变
```

评论区精华

1. gemini-code-assist[bot] 指出两个潜在风险：
   - 在 get_multimodal_data_bounds 中直接将 Python set 传入 torch.as_tensor 会引发类型错误，建议转换为 list。
   - 在 _get_chunked_embedding_by_item 中使用原始占位符计数分割 embedding，若编码器返回不同 token 数会崩溃，建议按比例缩放分割尺寸。
   - 结论 *：这些是回滚前 #25910 中的问题，回滚后相应代码被移除，风险自然消除。
2. 团队确认回滚：yhyang201（#25910 作者）在评论区回复"同意回滚"，并批准了该 PR。
3. CI 状态：amd-bot 报告 CI 失败，但 PR 作者认为非本 PR 引入。

- set 类型传入 torch.as_tensor 的潜在错误 (correctness): 该函数随回滚被移除，风险自动消失。
- embedding 分割尺寸不匹配的崩溃风险 (correctness): 该函数随回滚被移除，回到原先不依赖占位符计数的逐图像拼接路径。
- 原 PR 作者同意回滚 (other): 团队达成一致，批准合入。

风险与影响

- 风险：
  1. 功能特性退化：失去了跨请求 ViT 调用合并的批处理优化，在有多张图像的请求混合场景中视觉编码吞吐可能下降。但原有逐请求路径性能稳定，不影响正确性。
  2. 冲突区域的完整性：回滚时保留了 #26167 中 _can_skip_pre_embed_feature_move 函数，但其调用点需确认——旧路径中已有引用，无遗漏。
  3. AMD CI 稳定性：回滚后 AMD CI 已通过，但之前长期被阻塞；需关注是否存在其他间接依赖。
  4. 无测试覆盖：PR 未添加针对修复的回归测试，后续如有重提批处理优化时须补充。
- 影响：
  - 用户影响：AMD 平台用户恢复使用 VLM 模型，无其他用户可见变化。吞吐量从 870 token/s 恢复至 ~2700 token/s。
  - 系统影响：解除 main 分支的 CI 阻塞状态，恢复 AMD CI 的正常流水线。

- 团队影响：维护者需重新规划 #25910 的正确修复方案，并确保涵盖 MiniCPM-V-2.6 和 Qwen2.5-VL 等更多模型。
- 风险标记：功能特性退化，AMD CI 恢复

关联脉络

- PR #25910 vit optimization: 被本 PR 回滚的原始 PR，引入了跨请求 ViT 调用合并优化但导致 AMD CI 崩溃与性能回退。
- PR #26167 [VLM] feat: replace small H2D calls with a single one for qwen-vl: 回滚时产生冲突的唯一后续 PR，其新增的 _can_skip_pre_embed_feature_move 函数被保留，其余变更（涉及 _get_chunked_prefill_embedding 内部）被回滚覆盖。
- PR #26117 相关后续补丁（未直接冲突）：PR body 提到 #26117 等没有修改 _get_chunked_prefill_embedding 内部，因此回滚自动干净合并。