

PR #39145 完整报告

sgl-project/sglang

[Session] Fix image append positions and parent metadata

合并时间: 2026-09-14 12:06

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39145>

执行摘要

- 一句话: 修复 session 追加图像时 M-RoPE 位置缺失与 parent 元数据被篡改
- 推荐动作: 值得精读。该 PR 是「session 状态保存 + 多模态位置编码」交叉领域的典型正确性修复, 核心亮点有三个: 一是用 `dataclasses.replace` 副本隔离 parent 元数据, 二是用「旧媒体前缀 + 文本坐标 + 新媒体偏移」的方式补齐 M-RoPE 位置表, 三是「清空 stale 位置交还调度器」的失效兜底策略。后两点可作为后续处理位置敏感输入拼接的参考模式; 建议同时关注其测试设计——用独立 `cache_salt` 的全历史重放做参照, 是验证 session 一致性的高信噪比做法。

功能与动机

PR body 明确指出: "Appending an image after a generated reply can leave session positions incomplete and modify the saved parent's metadata. Stale positions can also prevent scheduler recomputation." 即 session 中先出回复、再追加图像时, 位置表缺了回复区间, 且 parent 的元数据会被污染; 残留的过期位置还会让调度器无法重算, 属于多模态 session 场景下的正确性缺陷。

实现拆解

实现按以下步骤拆解:

1. 变更入口路由: `Req.extend_image_inputs` 在 `python/sglang/srt/managers/schedule_batch.py` 中新增分支——`self.session is not None` 时转交 `_extend_session_image_inputs`, 非 session 请求保持原有 merge 路径, 避免影响常规多模态推理。
2. 强制重建 fill_ids: `_extend_session_image_inputs` 将 `self.full_untruncated_fill_ids` 重置为 `array("q")`。注释说明 padding 可能改变 token 数值但不改变数量, 因此必须让后续 `_refresh_fill_ids` 走全量重建, 防止旧 token 值残留。
3. 保护 parent 元数据: 当已有 `multimodal_inputs` 时用 `dataclasses.replace(self.multimodal_inputs)` 生成副本再做合并, 确保 sibling 分支、abort 重进的请求不会篡改已保存 session 父节点的媒体元数据。
4. 拼接 M-RoPE 位置表: 取新图像 `mrope_positions` 与 `origin_input_ids` 长度推导 `prefix_len`, 从旧位置表截取前缀坐标; 用 `torch.arange` 生成回复 / 文本区间的坐标 (从 `next_position` 起连续编号); 新图像坐标整体偏移 `next_position + text_len` 后 `torch.cat` 拼接, 形成「旧媒体 | 回复文本 | 新媒体」的完整位置表。

5. stale 位置失效策略：合并前将现有输入的 `mrope_positions`、`mrope_position_delta` 置 `None`（注释明确「或让调度器计算缺失位置」），合并后统一写回新拼接的位置表与 `mrope_position_delta`，并清空 `mrope_position_delta_repeated_cache`，避免旧缓存被复用。
6. 测试与 CI 配套：`test/registered/sessions/test_session_control.py` 新增 `TestSessionControlMrope`，用 Qwen2.5-VL-3B-Instruct 起流式 session 服务；`test_image_append` 覆盖普通追加、追加后从首轮分支、流式追加三类场景，均以独立 `cache_salt` 的全历史重放为参照断言 `output_ids` 完全一致；同时把 CUDA CI 注册时间从 102 秒上调到 137 秒以匹配新增用例耗时。

关键文件：

- `python/sglang/srt/managers/schedule_batch.py`（模块 调度器；类别 source；类型 core-logic；符号 `extend_image_inputs`, `_extend_session_image_inputs`）：核心修复所在：新增 `_extend_session_image_inputs` 处理 session 追加图像的 M-RoPE 位置拼接、`fill_ids` 重建与 parent 元数据保护，并让 `extend_image_inputs` 在 session 场景路由到新逻辑。
- `test/registered/sessions/test_session_control.py`（模块 会话测试；类别 test；类型 test-coverage；符号 `TestSessionControlMrope`, `setUpClass`, `tearDownClass`, `post`）：新增 `TestSessionControlMrope` 端到端回归：以 Qwen2.5-VL-3B 覆盖普通追加、sibling 分支、流式追加三种场景，并以独立全历史重放断言 `output_ids` 一致，是验证本修复的关键依据。

关键符号：`extend_image_inputs`, `_extend_session_image_inputs`,
`TestSessionControlMrope.test_image_append`

关键源码片段

`python/sglang/srt/managers/schedule_batch.py`

核心修复所在：新增 `_extend_session_image_inputs` 处理 session 追加图像的 M-RoPE 位置拼接、`fill_ids` 重建与 parent 元数据保护，并让 `extend_image_inputs` 在 session 场景路由到新逻辑。

```
def _extend_session_image_inputs(self, image_inputs):
    """在 session 中追加图像，保持已保存 session 与位置历史一致。"""
    # Padding 可能改变 token 数值但不改变数量，置空 fill_ids 强制后续走全量重建。
    self.full_untruncated_fill_ids = array("q")
    if self.multimodal_inputs is not None:
        # 分支与 abort 场景必须保留 parent 的 metadata，这里用 replace 生成副本再合并。
        self.multimodal_inputs = dataclasses.replace(self.multimodal_inputs)

    positions = image_inputs.mrope_positions
    if positions is not None:
        # origin_input_ids 比新位置表多出的部分，正是回复文本与追加文本的区间。
        prefix_len = len(self.origin_input_ids) - positions.shape[1]
        prefix = (
            self.multimodal_inputs.mrope_positions
```

```

        if self.multimodal_inputs is not None
        else None
    )
    if prefix is None:
        prefix = positions.new_empty((3, 0))
    # 只保留旧媒体坐标中的有效前缀，丢弃可能过期的尾部。
    prefix = prefix[:, :prefix_len]
    next_position = prefix.max() + 1 if prefix.numel() else 0
    text_len = prefix_len - prefix.shape[1]
    # 回复 / 追加文本的坐标从 next_position 起连续编号，铺满 3 个维度。
    text_positions = (
        torch.arange(
            text_len, dtype=positions.dtype, device=positions.device
        ).expand(3, -1)
        + next_position
    )
    # 新图像的媒体坐标整体右移，避免与旧媒体和文本坐标重叠。
    positions = torch.cat(
        [prefix, text_positions, positions + next_position + text_len], dim=1
    )

if self.multimodal_inputs is None:
    self.multimodal_inputs = image_inputs
else:
    # 先清空旧 delta，允许调度器用上面的完整表或自行重算缺失位置。
    self.multimodal_inputs.mrope_positions = None
    self.multimodal_inputs.mrope_position_delta = None
    self.multimodal_inputs.merge(image_inputs)

# 旧缓存可能基于过期位置表生成，必须一并失效。
self.multimodal_inputs.mrope_position_delta_repeated_cache = None
if positions is not None:
    self.multimodal_inputs.mrope_positions = positions
    self.multimodal_inputs.mrope_position_delta = (
        positions.max() + 1 - positions.shape[1]
    ).reshape(1, 1)

```

test/registered/sessions/test_session_control.py

新增 **TestSessionControlMrope** 端到端回归：以 Qwen2.5-VL-3B 覆盖普通追加、sibling 分支、流式追加三种场景，并以独立全历史重放断言 **output_ids** 一致，是验证本修复的关键依据。

```

def test_image_append(self):
    # 生成红 / 蓝两张 112x112 PNG，以 data URL 的形式传给服务端。
    images = []
    for color in ("red", "blue"):
        buffer = io.BytesIO()
        Image.new("RGB", (112, 112), color).save(buffer, format="PNG")
        images.append(
            "data:image/png;base64," + base64.b64encode(buffer.getvalue()).decode()
        )

```

```

    )
    prompt = (
        "<lim_startl>user\n<lvision_startl><limage_padl><lvision_endl>"
        "What color is this image?<lim_endl>\n<lim_startl>assistant\n"
    )
    sampling = dict(
        temperature=0, max_new_tokens=8, ignore_eos=True,
        skip_special_tokens=False, no_stop_trim=True,
    )
    for streaming in (False, True):
        with self.subTest(streaming=streaming):
            salt = uuid.uuid4().hex
            sid = self.post(
                "/open_session", capacity_of_str_len=4096, streaming=streaming
            )
            try:
                first = self.post(
                    "/generate", text=prompt, image_data=images[:1],
                    session_params={"id": sid}, cache_salt=salt,
                    sampling_params=sampling,
                )
                self.assertEqual(len(first["output_ids"]), 8)
                suffix = "<lim_endl>\n" + prompt
                # regular 模式追加后也覆盖从首轮再开分支的 sibling 场景。
                for image in images[1:] + (images[:1] if not streaming else []):
                    # 独立 cache_salt 避免 reference 请求先预热 session 的缓存。
                    reference = self.post(
                        "/generate",
                        text=prompt + first["text"] + suffix,
                        image_data=[images[0], image],
                        cache_salt=uuid.uuid4().hex,
                        sampling_params=sampling,
                    )
                    appended = self.post(
                        "/generate",
                        text=suffix,
                        image_data=[image],
                        session_params={"id": sid, "rid": first["meta_info"]["id"]},
                        cache_salt=salt,
                        sampling_params=sampling,
                    )
                    self.assertEqual(len(appended["output_ids"]), 8)
                    # session 追加与全历史重放的 token 必须完全一致。
                    self.assertEqual(appended["output_ids"], reference["output_ids"])
            finally:
                self.post("/close_session", session_id=sid)
            # 每次追加后服务端仍应健康可用。
            self.assertEqual(
                requests.get(self.base_url + "/health", timeout=10).status_code, 200

```

)

评论区精华

本 PR 没有留下 review 评论，讨论主要来自作者在 issue 中的 CI 归因说明：

"The inspected failures are unrelated to this change: dependency installation, test mocks, NPU out-of-memory, and AMD text-only kernel/disaggregation/logprob tests. None implicates the session image helper."

作者逐一排除了依赖安装、测试 mock、NPU 显存不足、AMD 文本内核 /PD/logprob 等失败项，均判定与本改动无关，并以 AMD 专项重跑（3 passed、2 skipped）和 H100 专项重跑（4 passed、1 disabled skip）作为佐证。整个 review 过程无实质技术争议，设计决策（副本保护 parent、stale 位置交还调度器）由作者单方定稿。

- CI 失败与本改动的关联性说明 (other)：失败被归因为环境与既有问题；H100 专项重跑 4 passed、1 disabled skip，AMD 专项重跑 3 passed、2 skip，验证了本改动的可靠性。

风险与影响

- 风险：风险集中在 `python/sglang/srt/managers/schedule_batch.py`：
- M-RoPE 拼接对长度假设敏感：`prefix_len = len(self.origin_input_ids) - positions.shape[1]` 依赖新图像位置表宽度与输入长度的精确对应，若未来 padding、图像占位符数量或多图拼接逻辑变化，位置表会整体错位。
- 浅拷贝的边界：`dataclasses.replace` 只做浅拷贝，嵌套的可变对象仍可能共享；当前实现靠显式清空 `mrope_positions`、`mrope_position_delta`、`mrope_position_delta_repeated_cache` 三个字段规避，后续新增字段时容易漏。
- 调度热路径变更：`extend_image_inputs` 被所有携带图像输入的请求调用，新增 session 分支属于热路径上的控制流调整，一旦回归影响面是所有 session 多模态请求。
- 测试覆盖单一模型：`TestSessionControlMrope` 仅覆盖 Qwen2.5-VL-3B 一个 M-RoPE 模型，其他 M-RoPE 系列模型未覆盖；且用例依赖真实 GPU 环境，无 CPU 单测兜底。
- 影响：影响范围集中在「流式 session + 多模态图像追加」这一功能交集：
- 用户侧：修复后，在 session 中先生成回复再追加新图像时输出 token 序列与独立重放完全一致，解决了此前回复区间位置缺失导致的错乱；同时 sibling 分支不再污染父节点元数据。
- 系统侧：session 追加图像请求会额外触发 `fill_ids` 全量重建与位置表拼接，带来少量重算开销；stale 位置失效策略让调度器可以按需重算，避免错误缓存复用。
- 团队侧：session-control 测试套件扩展为双模型覆盖（文本 + 多模态），CUDA CI 估算时长上调 35 秒；AMD stage-b 同步纳入新用例，两个平台均验证通过。
- 风险标记：会话核心路径变更，M-RoPE 位置拼接长度敏感，测试仅覆盖单一 M-RoPE 模型

关联脉络

- PR #39038 [Session] Work with PD and Fix empty continuations: 同一 session 控制功能线的修复，同样涉及 session 元数据与调度器交互，本 PR 的 parent 元数据保护思路与其一脉相承。

- PR #39120 Fix multimodal embedding cache retaining full batches through views: 同为多模态缓存 / 元数据生命周期修复，说明多模态输入对象的共享与拷贝问题在仓库中是持续关注点。