PR #35305 完整报告

sgl-project/sglang

[Kimi-K3] Fix "wrong grids" crash in DP-sharded vision preprocessing

合并时间：2026-08-23 09:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35305

执行摘要

- 一句话：修复 Kimi-K3 视觉 DP 分片下 grid 索引错位崩溃
- 推荐动作：值得精读。修复代码本身很薄，但包含两个可迁移的经验：一是 DP 分片下“索引域混淆”（shard-local vs global）是极易复发的 bug 类别，改动涉及多模态数据契约时应对索引空间做显式命名与映射；二是测试改进思路——用不同的输入值（不同 grid）暴露被相同值掩盖的回归，比新增更多断言更有效。建议后续 vision-DP 相关改动沿用该测试风格。

功能与动机

PR body 明确该变更修复 #34404 引入的回归：Kimi-K3 使用 cpu transport 与 DP-sharded vision 时，一个 batch 含 ≥2 张不同分辨率图片会以 ValueError: Kimi-K3 deferred GPU preprocessing produced wrong grids 崩溃。根因是延迟路径用 shard-local 位置读取 grid_thws_host，而该张量按全局图像位置索引，vision-DP 负载均衡会把全局索引重排后分发给各rank（如[2,1]），导致读错行。作者在4节点×8H100（TP32/EP32,cputransport）验证：三张不同分辨率图片的请求在 main 上崩溃、修复后正常完成。

实现拆解

变更入口：python/sglang/srt/models/kimi_k3.py 中 KimiK3ForConditionalGeneration.get_image_feature 内的延迟预处理闭包 materialize_item_features(image_indices)。

1. 定位索引域错位：闭包接收的 image_indices 是本 DP rank 应处理的全局图像位置，而 deferred_by_backend 分组产生的 indices 是 selected_items（rank 本地条目）内的 shard-local 位置；vision-DP 负载均衡会以重排切片（如 [2, 1]）调用 loader，两个索引域不再一致，grid_thws_host[indices] 因此读错行。
2. 修复映射：在分组循环入口新增 global_indices = [image_indices[index] for index in indices]；gpu backend 的 expected_grids 一致性校验与 patch_counts 特征切分两处读取 grid_thws_host 的位置改用 global_indices；写回 materialized[index] 仍用本地槽位，保持与后续 pixel_values.split(patch_counts) 的对应关系不变。
3. 强化测试：test/registered/unit/models/test_kimi_k3_vision.py 的 test_kimi_k3_preprocesses_only_dp_owner_images 将两张图的 grid 从相同的 [1, 1, 1] 改为 [1, 1, 1] 与 [1, 1, 2]，fake _gpu_preprocess_images 按各图 patch 数生成拼接 pixel_values，使旧实现读错行时必然触发 wrong grids 或长度不匹配校验；新断言验证

one.shape == (2, 2) 与 (one == 1.0).all()，同时补充函数 docstring。

4. 配套调整：python/sglang/kernels/ops/diffusion/__init__.py 的 _EXPORTS 新增 load_extension_with_recovery 包级导出，test_cpp_extension_loader.py 相应改为从 sglang.kernels.ops.diffusion 导入。该改动与主修复无逻辑耦合，讨论中未说明动机，属导出契约整理（字符串懒导出，风险低）。

验证与部署配套：4 节点 × 8 H100（TP32/EP32, cpu transport）手工验证修复生效；speed tests 声明无影响；无文档、schema 或部署配置改动。

关键文件：

- python/sglang/srt/models/kimi_k3.py（模块 视觉预处理；类别 source；类型 data-contract；符号 materialize_item_features, get_image_feature）：主修复文件：在延迟 GPU 预处理闭包 materialize_item_features 中新增 global_indices 映射，将 grid_thws_host 的读取从 shard-local 位置改为全局图像位置，修复 DP 分片下的 wrong grids 崩溃。
- test/registered/unit/models/test_kimi_k3_vision.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_kimi_k3_preprocesses_only_dp_owner_images）：测试强化：用不同 grid（[1, 1, 1] 与 [1, 1, 2]）构造两张图，使旧实现读错行时必然触发校验失败；旧测试用相同 grid 掩盖了回归。
- python/sglang/kernels/ops/diffusion/__init__.py（模块 内核导出；类别 infra；类型 infrastructure；符号 _EXPORTS, load_extension_with_recovery）：配套基础设施：在 _EXPORTS 中注册 load_extension_with_recovery 的包级导出，使测试可从 sglang.kernels.ops.diffusion 根导入。
- python/sglang/multimodal_gen/test/unit/test_cpp_extension_loader.py（模块 扩展加载；类别 test；类型 test-coverage）：配套测试更新：导入路径从 ext.loader 子模块改为包级入口，与 _EXPORTS 导出调整保持一致。

关键符号：materialize_item_features, KimiK3ForConditionalGeneration.get_image_feature, test_kimi_k3_preprocesses_only_dp_owner_images, load_extension_with_recovery

关键源码片段

python/sglang/srt/models/kimi_k3.py

主修复文件：在延迟 GPU 预处理闭包 materialize_item_features 中新增 global_indices 映射，将 grid_thws_host 的读取从 shard-local 位置改为全局图像位置，修复 DP 分片下的 wrong grids 崩溃。

```
# 延迟预处理分组循环：indices 是 selected_items（本 DP rank 的本地条目）内的
# shard-local 位置，而 grid_thws_host 按全局 batch 位置索引。vision-DP 下负载
# 均衡会给每个 rank 下发重排后的全局索引切片（例如 [2, 1]），所以必须先经
# image_indices 把本地位置映射回全局位置，再读取 grid 行。
for backend, indices in deferred_by_backend.items():
    group_items = [selected_items[index] for index in indices]
    group_configs = [deferred[index] for index in indices]
    # 修复点：建立 shard-local -> global 的索引映射，供下方两处读取使用。
    global_indices = [image_indices[index] for index in indices]
```

```
    first_config = group_configs[0]

    if backend == "gpu":
        # GPU 延迟预处理：先在 GPU 上实际跑一遍得到 produced_grids，
        # 再与全局 grid 行做一致性校验，任何错位都会在这里暴露。
        pixel_values, produced_grids = _gpu_preprocess_images(
            [item.feature for item in group_items],
            [config.resize_config for config in group_configs],
            image_scale,
            image_bias,
            self.vision_tower.patch_size,
            to_chw=lambda image: to_chw_uint8(image, device=device),
            post_resize=lambda x: fill_transparent_bg(
                x, first_config.transparent_bg_config
            ),
        )
        # 修复前误用 indices（shard-local）读取，多图不同分辨率时必然错行。
        expected_grids = grid_thws_host[global_indices]
        if not torch.equal(produced_grids.cpu(), expected_grids):
            raise ValueError(
                "Kimi-K3 deferred GPU preprocessing produced wrong grids"
            )
    elif backend == "cpu":
        pixel_values = materialize_kimi_k3_cpu_features(
            group_items, self._encoder_image_processor
        )
    else:
        raise ValueError(
            f"Unsupported Kimi-K3 deferred preprocessing backend: {backend}"
        )

    # 每个图像的 patch 数按全局 grid 计算，用于从拼接的 pixel_values 切回各图特征。
    patch_counts = [
        int(grid_thws_host[index].prod().item())
        for index in global_indices
    ]
    if sum(patch_counts) != pixel_values.shape[0]:
        raise ValueError(
            "Kimi-K3 deferred feature length does not match image grids"
        )
    # 注意写回仍用本地槽位 index，保持与 materialized 的分布一致。
    for index, feature in zip(
        indices, pixel_values.split(patch_counts), strict=True
    ):
        materialized[index] = feature
```

test/registered/unit/models/test_kimi_k3_vision.py

测试强化：用不同 grid（[1, 1, 1] 与 [1, 1, 2]）构造两张图，使旧实现读错行时必然触发校验失败；旧测试用相同 grid 掩盖了回归。

```
# 关键测试加强：两张图使用不同 grid（[1, 1, 1] 与 [1, 1, 2]）。
# 旧版测试用两个完全相同的 grid，出错时读到的那行与期望值一致，掩盖了回归。
grids = [[1, 1, 1], [1, 1, 2]]
patch_counts = [grid[0] * grid[1] * grid[2] for grid in grids]
items = [
    MultimodalDataItem(
        modality=Modality.IMAGE,
        offsets=[(index, index)],
        feature=torch.full((3, 2, 2), index, dtype=torch.uint8),
        model_specific_data={
            "image_grid_thw": torch.tensor([grids[index]]),
            DEFERRED_PREPROCESSING_KEY: deferred_config,
        },
    )
    for index in range(2)
]
calls = []


def fake_preprocess(images, resize_configs, *args, **kwargs):
    # fake 预处理按每张图各自的 patch 数生成拼接后的 pixel_values，
    # 若 patch_counts 按错误的 grid 切分，长度校验会立刻失败。
    ids = [int(image[0, 0, 0]) for image in images]
    calls.append(ids)
    pixel_values = torch.cat(
        [torch.full(size=(patch_counts[i], 2), fill_value=float(i)) for i in ids]
    )
    return pixel_values, torch.tensor([grids[i] for i in ids])
```

评论区精华

本 PR 无 review 评论，仅 3 条 issue 评论，三个 reviewer（mickqian ×2、mmangkad）均 APPROVED：

- fullyz："@mickqian PTAL. This fixes a regression from #34404."——直接点名回归来源，请求复审。
- mickqian："/tag-and-rerun-ci"——触发 CI 重跑。
- fullyz："@mickqian CI failures seem unrelated to this PR. Is it good to merge?"——CI Extra 失败后作者判断与本 PR 无关，询问是否可合入；随后合入。

值得注意：内核导出与导入路径的两处配套改动没有讨论记录，说明被视为低风险整理；PR 自身带 bypass-fastfail 标签。

- 回归来源定位（#34404）（other）：三位 reviewer（mickqian ×2、mmangkad）均 APPROVED，无 review 评论，修复方案被接受并合入。

- CI 失败是否阻塞合入 (question): 作者判断失败与本 PR 无关，PR 仍被合入；多分辨率 batch 的覆盖主要依赖强化后的单测与手工验证。

风险与影响

- 风险：
  1. 索引域对齐依赖：修复正确性依赖 image_indices 与 selected_items 一一对应；同文件测试 test_kimi_k3_rejects_aggregated_items 已固化“一个 item 恰含一张逻辑图”的契约，但上游（如 EPD encode server）若改变聚合语义，需同步复核该映射。
  2. 覆盖范围：本次仅修 gpu backend 延迟路径；cpu backend 走 materialize_kimi_k3_cpu_features 不读 grid_thws_host，不受影响，但未来若引入类似基于全局索引的读取需复用同一映射思路。
  3. CI 覆盖缺口：PR Test (Extra) 运行失败，作者判定无关后合入，多分辨率 batch 场景目前主要依赖强化后的单测与一次手工验证（4 节点 × 8 H100）。
  4. 导出面变更：sglang.kernels.ops.diffusion 新增 load_extension_with_recovery 包级导出并同步调整测试导入路径；若 ext.loader 存在循环依赖会在包导入时暴露，当前字符串懒导出设计下风险较低。
  5. 回归面：改动仅 7 行源码增量且集中在一个闭包内，本地预处理与单图场景行为完全不变。 - 影响：用户影响：修复 Kimi-K3 在 vision-DP + cpu transport + 同批多分辨率图片组合下的服务崩溃，这是真实部署会踩到的配置（作者在 4 节点 × 8 H100 复现并验证）；其他配置与模型不受影响。 系统影响：无性能影响（PR 声明 speed tests 无影响），源码增量 +7/−2，改动面极小。 团队影响：high priority 标签 + 快速合入，说明该问题影响实际用户；测试强化（不同 grid）为后续 DP 视觉回归提供了有效基线，避免“相同值掩盖 bug”的测试盲区再次出现。
- 风险标记：回归修复（#34404），索引域对齐依赖，CI Extra 未通过，附带内核导出变更

关联脉络

- PR #34404 引入该回归的 delayed preprocessing 变更（标题未提供）：PR body 与评论明确点名 #34404 为本回归的来源：延迟 GPU 预处理路径以 shard-local 位置读取按全局索引组织的 grid_thws_host。
- PR #34490 [AMD] Add Radix-4 MoE top-k router kernel for Kimi-K3 routing: 同一 Kimi-K3 模型线的 kernels 侧改动，且本 PR 也包含对 sglang.kernels 导出面的调整；子系统不同，属于弱关联。
- PR #35508 [NPU] [DOC] Add Ascend NPU (A3) recipe to the Kimi-K3 cookbook: 同一 Kimi-K3 模型线的部署文档配套，反映该模型支持仍在多线演进。