PR #33349 完整报告

sgl-project/sglang

[Perf] Speed up the Kimi-K2.5 vision path and match PIL bicubic in the GPU resize

合并时间：2026-08-04 12:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33349

执行摘要

- 一句话：K2.5 视觉路径提速并修正 GPU 缩放与 PIL 对齐
- 推荐动作：值得精读。该 PR 是 " 性能优化 + 数值对齐 " 结合的范本：融合 kernel、消除同步点、绕过 tokenizer 往返等手法可直接迁移到其他 VLM 预处理路径。重点看三处：_resize_bicubic_if_needed 对 PIL 语义的逐点拆解（antialias + uint8 量化）、verify_k25_equivalence.py 用数值证据支撑重写等价的方法论、以及 dtype 门控和 host 侧 grid_thws 这类 " 不做多余的事 " 的契约设计。注意该 PR 没有独立评审，阅读时建议对 uint8 强校验与处理器并发时序保持警惕，并确认上游解码路径均满足新契约。

功能与动机

PR body 陈述了性能与正确性双动机。性能上，K2.5 视觉路径存在多处可见开销：每 encoder block 一次 view_as_complex 往返、每 forward 一次 max_seqlen 设备同步，以及 " decoding to text, splicing thousands of repeated <|media_pad|> strings and re-tokenizing" 的冗余 tokenizer 往返；body 给出实测 QK RoPE 27 层 12544 tokens 从 4.604ms 降到 0.649ms。正确性上，KimiGPUProcessorWrapper 绕过 HF media processor 用 F.interpolate(mode='bicubic') 在 GPU 缩放，而 "PIL's bicubic widens its kernel support by the scale factor, i.e. it always antialiases on downscale"，且 "PIL returns uint8"——旧路径把带过冲的浮点直接喂给归一化。实测误差表显示 photo 类图像 mean/max err 从 3.79/34.5 降到 0.15/2.0，并明确 "This changes model inputs on the K2.5 GPU preprocessing path — toward the reference processor, not away from it."

实现拆解

1. GPU resize 对齐 PIL bicubic（kimi_k25.py 处理器）：新增 _resize_bicubic_if_needed，在 F.interpolate(mode='bicubic', align_corners=False) 上补 antialias=True 以复现 PIL 缩小时的隐式抗锯齿，并对结果 round_().clamp_(0.0, 255.0) 还原 uint8 像素语义；目标尺寸一致时只做 float 转换以省掉无谓重采样。_ensure_chw_rgb 增加 uint8 强校验（拒绝已归一化浮点图），_process_single_image 与 _resize_images_by_source_shape 统一改走新函数。
2. 融合 kernel 消除往返（kimi_k25.py 模型侧）：apply_rope 新增分支——收到 PreparedInplaceComplexRoPE 元组时调用 apply_fused_qk_complex_rope_inplace（来自 sglang.kernels.ops.attention.vision_rope），否则保留 view_as_complex 可移植路径。MoonViT3dEncoder 通过 use_fused_rope 门控（仅 CUDA 且非 RL 确定性模式且 dtype

为 fp16/bf16 时开启），避免发布未测试的 dtype 分支。预处理侧 normalize_and_patchify 把 pad -> /255 -> (x - mean) * inv_std -> reshape/permute 整条链折叠为一次仿射（scale = 1/(255*std)，bias = -mean/std）。

3. 消除设备同步（vision.py、kimi_k25.py）：prepare_vision_attention_metadata 新增可选参数 max_seqlen，调用方把 MoonViT3dEncoder 中已算好的 host 整数传入，避免 attention 后端每次 forward 再做一次 device-to-host 同步；grid_thws 保持 host 端——MoonViT3d 只把它当形状元数据（.tolist()），不再 .to(device)，该契约与 encoder-DP 路径在 mm_utils 中的既有约定一致。
4. 跳过 tokenizer 往返并收紧占位符契约（kimi_k25.py、kimi_common.py、kimi_vl.py、base_processor.py）：_expand_image_token_ids 用 np.repeat 在数组域直接展开原始 token id，配合 preserve_processor_input_ids 让 _cpu_call 的 CPU fallback 也保留请求 token；kimi_common.py 新增静态方法 count_image_placeholders（文本 prompt 返回 None），kimi_vl.py 的 process_mm_data_async 增加图片数量与占位符 1:1 校验并在不匹配时抛错。
5. 收尾与测试配套：mm_utils.py 抽出 concat_or_single，run_dp_sharded_mrope_vision_model 的三处 concat 与 mm_projection_auto 共用（单图请求不再拷贝 embedding）；mm_projection_auto 改为返回打包的 2D 特征；tpool_patch_merger 迁入 kimi_vl_moonvit.py 并在 t == 1 时跳过 temporal mean（与求均值逐位一致）。测试侧：test_kimi_k25.py 新增 8 组用例（PIL bicubic 对齐、占位符展开 / 校验、CPU fallback 保留 token、uint8 守卫、单帧池化等价、打包投影契约、grid_thws 的 meta 设备守卫），test_kimi_k3_prerequisite_ops.py 新增 test_vision_rope_inplace，并新增需 GPU 的手工脚本 test/manual/vlm/verify_k25_equivalence.py 证明两类重写与参考实现数值等价。

关键文件：

- python/sglang/srt/multimodal/processors/kimi_k25.py（模块 预处理；类别 source；类型 core-logic；符号 _expand_image_token_ids, _resize_bicubic_if_needed, _grid_thw_from_resize_config, _to_cuda_chw）：K2.5 GPU 预处理主路径：PIL bicubic 对齐、占位符展开、uint8 契约均在此落地，是正确性修复与大部分性能收益的载体
- python/sglang/srt/models/kimi_k25.py（模块 视觉模型；类别 source；类型 core-logic；符号 apply_rope, MoonViT3dEncoder.forward, mm_projection_auto, tpool_patch_merger）：融合 QK RoPE 集成与 max_seqlen 上提、grid_thws host 契约、投影打包——模型侧核心性能改动
- test/registered/unit/models/test_kimi_k25.py（模块 单测；类别 test；类型 test-coverage；符号 test_kimi_resize_tracks_the_checkpoint_processors_pil_bicubic, test_kimi_resize_is_a_dtype_only_cast_when_already_at_target, test_kimi_expands_one_placeholder_per_image_from_existing_ids, test_kimi_expansion_rejects_a_placeholder_count_mismatch）：核心回归防线：PIL bicubic 对齐、占位符展开 / 校验、CPU fallback、uint8 守卫等新增 8+ 用例，并验证 grid_thws 的 meta 设备守卫
- python/sglang/srt/models/kimi_vl_moonvit.py（模块 共享组件；类别 source；类型 refactor；符号 tpool_patch_merger）：tpool_patch_merger 迁入成为 Kimi VL/K2.5 共

享组件，t==1 跳过 mean 行为与求均值逐位一致

- test/manual/vlm/verify_k25_equivalence.py（模块 验证脚本；类别 test；类型 test-coverage；符号 reference_preprocess, check_patchify, check_padded_value_is_not_zero, reference_rope）：手工等价性验证脚本：证明 normalize+patchify 与融合 RoPE 重写与参考实现数值等价（RoPE 逐位一致）
- python/sglang/srt/multimodal/processors/kimi_common.py（模块 公共逻辑；类别 source；类型 core-logic；符号 count_image_placeholders）：新增 count_image_placeholders 静态方法，定义占位符计数的共享契约（文本 prompt 返回 None）
- python/sglang/srt/multimodal/mm_utils.py（模块 公共工具；类别 source；类型 refactor；符号 concat_or_single）：concat_or_single 抽成公共工具，DP encoder 与投影器共用，单图路径不再拷贝 embedding
- test/registered/kernels/ops/test_kimi_k3_prerequisite_ops.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_vision_rope_inplace, test_vision_rope_inplace_rejects_non_complex_frequencies）：为本 PR 实际落地的原地 RoPE kernel 提供 bf16/fp16 精度与 non-complex 输入校验用例
- python/sglang/srt/multimodal/processors/kimi_vl.py（模块 处理器；类别 source；类型 core-logic；符号 process_mm_data_async）：KimiVL 同步获得 1:1 占位符校验，避免静默错配
- python/sglang/srt/layers/attention/vision.py（模块 注意力层；类别 source；类型 core-logic；符号 prepare_vision_attention_metadata）：prepare_vision_attention_metadata 增加可选 max_seqlen，消除每 forward 的 H2D 同步
- python/sglang/srt/multimodal/processors/base_processor.py（模块 基类处理器；类别 source；类型 core-logic；符号 preserve_processor_input_ids）：preserve_processor_input_ids 开关使 base 类跳过重复重建，是 tokenizer 往返绕过得以成立的前提

关键符号：_expand_image_token_ids, _resize_bicubic_if_needed, _grid_thw_from_resize_config, _to_cuda_chw, _prepare_input_ids, _gpu_call, _cpu_call, apply_rope, prepare_fused_qk_complex_rope_inplace, apply_fused_qk_complex_rope_inplace, mm_projection_auto, tpool_patch_merger, count_image_placeholders, concat_or_single, prepare_vision_attention_metadata, normalize_and_patchify

关键源码片段

python/sglang/srt/multimodal/processors/kimi_k25.py

K2.5 GPU 预处理主路径：PIL bicubic 对齐、占位符展开、uint8 契约均在此落地，是正确性修复与大部分性能收益的载体

```
def _resize_bicubic_if_needed(
    image: torch.Tensor, target_height: int, target_width: int
) -> torch.Tensor:
```

```
    """复现 checkpoint 处理器的 PIL.Image.resize(..., BICUBIC) 语义。

    两个关键差异要同时处理：PIL 的 bicubic 在缩小时会按缩放因子加宽核支撑、
    总是做 antialias，而 F.interpolate 只有在 antialias=True 时才等价；
    另外 PIL 返回 uint8，所以这里要 round 并 clamp 回 [0, 255]，
    否则 bicubic 过冲会带着未量化的浮点值一起进入归一化。
    """
    image = image.float()
    # 目标尺寸已一致时只做 dtype 转换，不触发无谓的重采样
    if image.shape[-2:] == (target_height, target_width):
        return image
    return (
        F.interpolate(
            image,
            size=(target_height, target_width),
            mode="bicubic",
            align_corners=False,
            antialias=True,
        )
        .round_()
        .clamp_(0.0, 255.0)
    )


def _expand_image_token_ids(
    input_ids: Union[List[int], torch.Tensor],
    image_token_id: int,
    image_token_counts: List[int],
) -> torch.Tensor:
    """直接展开原始 token id，跳过"解码 -> 拼接文本 -> 重新 tokenize"的往返。

    语义与 BaseMultimodalProcessor._expand_input_ids 保持一致（单测双向钉死），
    这里全程留在数组域，用一次 np.repeat 完成展开，
    避免为每个请求创建临时列表。
    """
    if isinstance(input_ids, torch.Tensor):
        input_ids = input_ids.detach().flatten().cpu().numpy()
    input_ids = np.asarray(input_ids, dtype=np.int64)

    placeholder_mask = input_ids == image_token_id
    placeholder_count = np.count_nonzero(placeholder_mask)
    if placeholder_count != len(image_token_counts):
        raise ValueError(
            f"Expected {len(image_token_counts)} image placeholder token(s), "
            f"found {placeholder_count}."
        )

    # 每个占位符位置替换为对应图片的 token 数，其余位置重复 1 次
    repeats = np.ones(input_ids.shape, dtype=np.int64)
```

```
    repeats[placeholder_mask] = image_token_counts
    return torch.from_numpy(np.repeat(input_ids, repeats)).unsqueeze(0)
```

python/sglang/srt/models/kimi_k25.py

融合 QK RoPE 集成与 max_seqlen 上提、grid_thws host 契约、投影打包——模型侧核心性能改动

```
def apply_rope(
    xq: torch.Tensor,
    xk: torch.Tensor,
    freqs_cis: torch.Tensor | PreparedInplaceComplexRoPE,
    x_shape=None,
) -> tuple[torch.Tensor, torch.Tensor]:
    """q/k RoPE 应用入口，同时支持融合与可移植两条路径。

    收到 PreparedInplaceComplexRoPE 元组时走融合的 CUDA 原地 kernel；
    否则退回 view_as_complex 的复数乘法路径（非 fp16/bf16 或非 CUDA）。
    """
    if isinstance(freqs_cis, tuple):
        return apply_fused_qk_complex_rope_inplace(xq, xk, freqs_cis)

    # 可移植路径：复数乘法后拆回实数，每层一次往返，开销高但无 JIT 依赖
    freqs_cis = freqs_cis.unsqueeze(-2) # ..., 1, head_dim/2
    xq_ = torch.view_as_complex(xq.float().view(*xq.shape[:-1], -1, 2))
    xk_ = torch.view_as_complex(xk.float().view(*xq.shape[:-1], -1, 2))
    xq_out = torch.view_as_real(xq_ * freqs_cis).flatten(-2)
    xk_out = torch.view_as_real(xk_ * freqs_cis).flatten(-2)
    return xq_out.type_as(xq), xk_out.type_as(xk)


class MoonViT3dEncoder(nn.Module):
    # 类级默认值保证用 __new__ 构造的单元测试实例也能直接 forward；
    # 实例的 use_fused_rope 由 __init__ 根据 CUDA 环境与 RL 确定性配置决定
    use_fused_rope = False

    def forward(self, hidden_states, grid_thws):
        rope_freqs_cis = self.rope_2d.get_freqs_cis(
            grid_thws=grid_thws, device=hidden_states.device
        )
        # 原地 kernel 是 q/k dtype 上的 JIT 模板，只有 fp16/bf16 有测试覆盖，
        # 其他 dtype 留在可移植路径，避免发布未经验证的 kernel 分支
        if self.use_fused_rope and hidden_states.dtype in (
            torch.float16,
            torch.bfloat16,
        ):
            rope_freqs_cis = prepare_fused_qk_complex_rope_inplace(rope_freqs_cis)

        # 先取 host 端 max_seqlen 再一次性搬到设备端，
        # 避免每个 encoder block 各自触发一次 device-to-host 同步
```

```
    sequence_lengths = grid_thws[:, 0] * grid_thws[:, 1] * grid_thws[:, 2]
    max_seqlen = int(sequence_lengths.max().item())
    sequence_lengths = sequence_lengths.to(
        device=hidden_states.device, dtype=torch.int32
    )
    lengths = torch.cat(
        (
            torch.zeros(1, dtype=torch.int32, device=hidden_states.device),
            sequence_lengths,
        )
    )
    cu_seqlens = lengths.to(hidden_states.device).cumsum(dim=0, dtype=torch.int32)

    # 显式传入已算好的 max_seqlen，attention 元数据不再重复同步
    forward_metadata = prepare_vision_attention_metadata(
        cu_seqlens,
        device=hidden_states.device,
        max_seqlen=max_seqlen,
    )

    for block in self.blocks:
        hidden_states = block(
            hidden_states,
            cu_seqlens,
            max_seqlen,
            rope_freqs_cis=rope_freqs_cis,
            forward_metadata=forward_metadata,
            sequence_lengths=sequence_lengths,
        )
    return self.final_layernorm(hidden_states)
```

评论区精华

该 PR 的 review_comments_count 为 0，没有独立评审评论；作者 hnyls2002 通过多轮 /rerun-test 驱动 CI（单卡 H100/5090、4-gpu、8-gpu 与 gb300 的视觉测试矩阵）后自行合入。虽然没有评审交锋，PR body 自带的高质量自证值得提炼：

- 对 PIL 残差的归因："Against PIL's float path the two agree to ~5e-3, so the resampling kernel now matches exactly and the residual above is only PIL's fixed-point arithmetic for uint8 images."
- 融合 RoPE 与 view_as_complex 参考实现 "bit-identical (max abs diff 0.00e+00)"，覆盖 bf16/fp16 x {1024, 4096, 37} tokens x {16, 8, 4} heads x {72, 128, 64} head dims。
- dtype 门槛的取舍说明："The in-place kernel is a JIT template on the q/k dtype; only fp16/bf16 are covered by tests, so other dtypes stay on the portable path."
- 测试设计特意用 "photo-like content, not pure noise" 构造输入：纯噪声下 antialias 差异会被掩盖，而照片类内容下 naive 与 PIL 偏差 > 20、修复后 <= 4 个 8-bit 级别。这些表述实际承担了 " 设计决策评审 " 的角色，可作为无评审 PR 的自证范式参考。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 模型输入语义变更（kimi_k25.py）：GPU 预处理从 " 无 antialias 的浮点 bicubic" 变为 " 仿 PIL 的 uint8 量化结果 "，这是有意的行为变更，但依赖旧数值的基准、特征缓存与回归测试需要同步校准。
  2. JIT kernel 覆盖范围（kimi_k25.py 模型侧）：apply_fused_qk_complex_rope_inplace 仅 fp16/bf16 有测试；use_fused_rope 还耦合了与视觉路径无关的 get_exec().deterministic.rl_on_policy_target 判断，若执行上下文行为变化，可能让未测试 dtype 误入融合路径。
  3. 处理器并发时序敏感：supports_mm_processor_concurrency（2 处理器 / 16 IO worker）依赖在 super().__init__ 之前构建 GPU wrapper 的时序——基类构造会 clone self._processor 给 worker 池，时序被破坏就会静默绕过 Kimi 的 GPU 预处理。
  4. uint8 强校验为破坏性变更：_ensure_chw_rgb 直接拒绝非 uint8 张量，传入已归一化浮点图的既有调用方会抛错，需要确认所有上游解码路径（nvJPEG、缓存张量）都满足 uint8 契约。
  5. 占位符 1:1 校验收紧：kimi_vl.py 与 _expand_image_token_ids 的计数校验会让以前被宽松接受的 prompt 直接失败。 - 影响：用户侧：Kimi-K2.5/K2.7 推理每张 1024x1024 图像约省 1.9ms（QK RoPE 6.8-7.1x，normalize+patchify 4.5-5.4x，attention metadata 每 forward 61.4us -> 39.4us），且模型输入更贴近 checkpoint 参考处理器，数值上与 HF 基线更一致。系统侧：prepare_vision_attention_metadata 新增可选参数，对所有既有调用方向后兼容；concat_or_single 替换 torch.cat 在单元素时不再拷贝，run_dp_sharded_mrope_vision_model 的 DP encoder 路径行为等价且有测试覆盖。团队侧：verify_k25_equivalence.py 建立了 " 重写必须证明与参考实现等价 " 的验证范式；count_image_placeholders 成为 Kimi VL 与 K2.5 共享的行为契约；vision_rope 的原地 kernel 同时是 Kimi-K3 的前置算子，具备后续复用价值。 - 风险标记：模型输入语义变更，JIT kernel 仅测 fp16/bf16，处理器并发时序敏感，uint8 校验破坏性变更，无独立 review

关联脉络

- PR #33367 fix: pi05 models does not apply scale factor for language embeddings: 同为多模态预处理与参考实现对齐的正确性修复（Pi0.5 语言嵌入缺失缩放 vs K2.5 GPU bicubic 不匹配），修复手法都是构造参考实现逐点对照；两者属不同模块（multimodal_gen vs srt 处理器），无文件交集。
- PR #33453 [diffusion] Restrict request-level quality to two validated tiers: lossless (default) and high: 都在收紧处理器行为契约：该 PR 把未验证档位收窄为两级，本 PR 把 uint8 输入与占位符 1:1 校验写入预处理路径，同属 " 减少静默错误输入 " 的演进方向。