PR #34766 完整报告

sgl-project/sglang

[Fix] Carry the backend on Kimi-K3 deferred preprocessing configs

合并时间：2026-08-14 04:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34766

执行摘要

- 一句话：修复 Kimi-K3 deferred 预处理配置缺失 backend 导致的崩溃
- 推荐动作：值得精读。该 PR 虽小，但展示了如何用类型化数据契约（frozen dataclass）替代裸 dict 来消除隐式字段依赖，将运行时 KeyError 前置为构造期错误，是一种可复用的防御性设计模式。同时 functools.partial 绑定公共字段的做法也值得借鉴。建议阅读 kimi_k3_image_processing.py 的 dataclass 定义与 models/kimi_k3.py 的消费端改造。

功能与动机

PR body 明确指出：Kimi-K3's tokenizer-side deferred path never wrote backend into its preprocessing config, so materialize_item_features raised KeyError: 'backend' and took down every TP rank once --mm-feature-transport started defaulting to cpu。因此需要让 backend 字段在配置构造时就必须存在，而不是在物化时才暴露缺失。

实现拆解

1. 新增 frozen dataclass 定义契约：在 python/sglang/srt/multimodal/kimi_k3_image_processing.py 中新增 @dataclass(frozen=True) 的 KimiK3DeferredPreprocessing，字段为 backend（Literal["gpu", "cpu"]）、image_mean、image_std、transparent_bg_config、resize_config。frozen 保证配置在跨 rank 传递中不可变，且构造时任何字段缺失都会立即报错，把运行时 KeyError 提前到配置创建阶段。
2. 改造 encoder 侧构造点：prepare_kimi_k3_encoder_inputs 不再拼装 common_deferred_config 字典，而是用 functools.partial(KimiK3DeferredPreprocessing, backend=..., image_mean=..., ...) 预绑定公共字段，每个 item 只需传入 per-image 的 resize_config。同时移除了 materialize_kimi_k3_cpu_features 中对 feature_layout == "raw" 的防御性检查，因为布局信息可从 item.feature 直接观察，且 dataclass 不再携带该字段。
3. 改造 tokenizer 侧构造点：python/sglang/srt/multimodal/processors/kimi_k3.py 的 KimiK3GPUProcessorWrapper.prepare_deferred 和 KimiK3ImageProcessor._build_deferred_output 同样改为返回 / 消费 functools.partial，并注释说明该路径只做 GPU deferred，因此 backend 固定为 "gpu"。
4. 消费端改为属性访问：python/sglang/srt/models/kimi_k3.py 的 materialize_item_features 将所有 config["xxx"] 改为 config.xxx，并增加 first_config.backend 的一致性检查，确保同 batch 内 deferred backend 不混用。

5. 测试配套更新：test_kimi_k25.py、test_kimi_k3_vision.py、test_kimi_k3_encoder_mode.py 三个测试文件从构造 dict 改为构造 KimiK3DeferredPreprocessing 对象，并新增对 config.backend == ["gpu", "gpu"] 和 resize_config["new_width"] 的断言，覆盖 tokenizer 侧 deferred 路径的 backend 传递。

关键文件：

- python/sglang/srt/multimodal/kimi_k3_image_processing.py（模块 多模态；类别 source；类型 core-logic；符号 KimiK3DeferredPreprocessing, prepare_kimi_k3_encoder_inputs, materialize_kimi_k3_cpu_features）：修复核心：新增 frozen dataclass KimiK3DeferredPreprocessing，将 deferred 配置从裸 dict 收敛为强类型契约，并改造 encoder 侧构造点与物化检查。
- python/sglang/srt/multimodal/processors/kimi_k3.py（模块 处理器；类别 source；类型 core-logic；符号 KimiK3GPUProcessorWrapper.prepare_deferred, KimiK3ImageProcessor._build_deferred_output）：tokenizer 侧 deferred 路径的修复点：prepare_deferred 原本不写 backend，现在固定 backend="gpu" 并返回 partial，_build_deferred_output 消费该 partial。
- python/sglang/srt/models/kimi_k3.py（模块 模型层；类别 source；类型 data-contract；符号 KimiK3ForConditionalGeneration.materialize_item_features）：消费端 materialize_item_features 从 dict 下标访问改为属性访问，并保持 backend 一致性校验，是数据契约落地的一环。
- test/registered/unit/models/test_kimi_k25.py（模块 测试；类别 test；类型 test-coverage；符号 test_kimi_k3_cpu_transport_defers_gpu_preprocessing）：覆盖 tokenizer 侧 deferred 路径，断言 backend 为 gpu 且 resize_config 正确传递，是回归测试的关键。
- test/registered/unit/models/test_kimi_k3_vision.py（模块 测试；类别 test；类型 test-coverage；符号 test_kimi_k3_preprocesses_only_dp_owner_images）：覆盖视觉 DP owner 物化路径，改用 dataclass 构造 deferred 配置，验证属性访问兼容性。
- test/registered/unit/disaggregation/test_kimi_k3_encoder_mode.py（模块 测试；类别 test；类型 test-coverage；符号 test_kimi_k3_epd_preprocess_preserves_raw_per_image_items, test_kimi_k3_epd_default_cpu_materialization_is_owner_only_and_exact）：覆盖 encoder 侧 EPD 预处理，断言改为属性访问，验证 backend 正确写入。

关键符号：KimiK3DeferredPreprocessing, prepare_kimi_k3_encoder_inputs, prepare_deferred, _build_deferred_output, materialize_item_features

关键源码片段

python/sglang/srt/multimodal/kimi_k3_image_processing.py

修复核心：新增 frozen dataclass KimiK3DeferredPreprocessing，将 deferred 配置从裸 dict 收敛为强类型契约，并改造 encoder 侧构造点与物化检查。

```
import functools
from dataclasses import dataclass
from typing import Literal, Optional
```

```
@dataclass(frozen=True)
class KimiK3DeferredPreprocessing:
    """延迟预处理的完整参数，由视觉 DP 属主在物化时使用。

    ``backend`` 是生产方的决策，无法从 item 中恢复；而特征布局仍可从
    ``item.feature`` 观察，因此这里不冗余保存。frozen 保证配置在跨 rank
    传递中不可变，任何字段缺失都会在构造时立即报错，而不是等到物化阶段
    才抛 ``KeyError``。
    """

    backend: Literal["gpu", "cpu"]
    image_mean: list[float]
    image_std: list[float]
    transparent_bg_config: Optional[dict]
    resize_config: dict


def prepare_kimi_k3_encoder_inputs(
    images, image_processor, *, use_gpu_preprocessing=False
):
    # 前置逻辑省略，直接看核心改动：
    # 用 partial 预绑定公共字段，每个 item 只需传入 per-image 的 resize_config
    deferred_preprocessing = functools.partial(
        KimiK3DeferredPreprocessing,
        backend="gpu" if use_gpu_preprocessing else "cpu",
        image_mean=list(media_proc_cfg["image_mean"]),
        image_std=list(media_proc_cfg["image_std"]),
        transparent_bg_config=media_proc_cfg.get("transparent_bg_config"),
    )
    item = MultimodalDataItem(
        modality=Modality.IMAGE,
        feature=to_chw_uint8(image) if use_gpu_preprocessing else image,
        model_specific_data={
            "grid_thws": grid_tensor,
            DEFERRED_PREPROCESSING_KEY: deferred_preprocessing(
                resize_config=resize_config
            ),
        },
    )
```

python/sglang/srt/multimodal/processors/kimi_k3.py

tokenizer 侧 deferred 路径的修复点：prepare_deferred 原本不写 backend，现在固定 backend="gpu" 并返回 partial，_build_deferred_output 消费该 partial。

```
# 该路径只做 GPU deferred：调用方由 _should_defer_gpu_preprocessing 把关，
# 且 staging 的是 CHW uint8 特征，所以 backend 固定为 "gpu"。
def prepare_deferred(self, text, images, original_input_ids=None):
    input_text = text[0] if isinstance(text, list) else text
    image_sizes = [_get_image_dimensions(image) for image in images]
```

```
    resize_configs = [
        navit_resize_config(
            width, height, self._patch_size, self._merge_kernel_size,
            self._in_patch_limit, self._patch_limit_on_one_side,
            self._fixed_output_tokens,
        )
        for width, height in image_sizes
    ]
    input_ids = self._prepare_input_ids(
        input_text, resize_configs, original_input_ids, image_sizes
    )
    deferred_preprocessing = functools.partial(
        KimiK3DeferredPreprocessing,
        backend="gpu",
        image_mean=list(self._image_mean),
        image_std=list(self._image_std),
        transparent_bg_config=self._transparent_bg_config,
    )
    return input_ids, resize_configs, deferred_preprocessing


def _build_deferred_output(self, base_output):
    input_ids, resize_configs, deferred_preprocessing = (
        self._processor.prepare_deferred(
            base_output.input_text,
            base_output.images,
            base_output.input_ids,
        )
    )
    # 每个 image 只传自己的 resize_config，其余公共字段已在 partial 中绑定
    item = MultimodalDataItem(
        modality=Modality.IMAGE,
        feature=to_chw_uint8(image),
        offsets=[offset],
        model_specific_data={
            "image_grid_thw": torch.tensor([grid_thw], dtype=torch.int64),
            DEFERRED_PREPROCESSING_KEY: deferred_preprocessing(
                resize_config=resize_config
            ),
        },
    )
```

python/sglang/srt/models/kimi_k3.py

消费端 materialize_item_features 从 dict 下标访问改为属性访问，并保持 backend 一致性校验，是数据契约落地的一环。

```
# 视觉 DP 属主物化时按 backend 分发；所有字段已改为属性访问。
# 由于 dataclass 是 frozen 的，这里天然保证每个 config 都有 backend。
first_config = deferred[0]
```

```
backend = first_config.backend
if any(config.backend != backend for config in deferred):
    raise ValueError("Kimi-K3 cannot mix deferred preprocessing backends")

if backend == "gpu":
    image_scale, image_bias = normalization_tensors(
        first_config.image_mean, first_config.image_std, device
    )
    pixel_values, _ = _gpu_preprocess_images(
        [item.feature for item in selected_items],
        [config.resize_config for config in deferred],
        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
        ),
    )
elif backend == "cpu":
    pixel_values = materialize_kimi_k3_cpu_features(
        selected_items, self._encoder_image_processor
    )
    pixel_values = pixel_values.to(device, non_blocking=True)
else:
    raise ValueError(
        f"Unsupported Kimi-K3 deferred preprocessing backend: {backend}"
    )
```

评论区精华

该 PR 没有代码 review 评论，主要验证过程集中在 issue 评论的 CI 重跑：作者多次触发 /rerun-test，覆盖 test_kimi_k25.py、test_kimi_k3_vision.py、test_kimi_k3_encoder_mode.py 和 test_kimi_k3_b300.py。第一轮 B300 *e2e* 失败后重试通过，最终所有相关测试均成功，说明修复在 CPU 单测与真实 GPU 拓扑上都得到验证。

- CI 测试重跑验证 (testing): 所有相关单测与 B300 *e2e* 最终通过，修复得到验证。

风险与影响

- 风险：
  1. 数据契约变更风险：model_specific_data 中该 key 的值从 dict 变为 dataclass，凡是直接构造该 dict 的外部扩展或序列化路径都会受影响。当前源码内两个构造点已同步更新，但需确认没有其他模块（如缓存、序列化）直接假定 dict 结构。
  2. 防御性检查移除：materialize_kimi_k3_cpu_features 删除了 feature_layout == "raw" 检查，若未来有调用方误传入 CHW uint8 特征，错误会延迟到 PIL/Tensor 类型检查时才暴露，排查难度略增。

3. 可序列化性：functools.partial 与 dataclass 一般可 pickle，但若后续跨进程传输需要确认无绑定局部对象导致序列化失败的风险。
4. 回归范围：本次改动同时触碰 Kimi-K2.5/K3 共用的 processor 代码，测试覆盖了相关路径，但真实流量下 deferred 与 non-deferred 混用场景仍需关注。 - 影响：影响范围集中在 Kimi-K3 多模态推理链路：修复了 --mm-feature-transport 默认 cpu 时所有 TP rank 崩溃的阻断性问题，使 deferred 预处理路径在默认配置下可用。对使用自定义 dict 构造 deferred 配置的第三方扩展属于 breaking change，但此类用法鲜见。团队内受益于类型化契约，后续新增字段会被编译器 / 解释器强制要求，减少同类 KeyError 回归。 - 风险标记：默认路径崩溃修复，数据契约变更，防御性检查移除

关联脉络

- 暂无明显关联 PR