

PR #39148 完整报告

sgl-project/sglang

[MM] Add flag to force Kimi image preprocessing onto CPU

合并时间: 2026-09-15 12:19

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39148>

执行摘要

- 一句话: 为 Kimi 新增 CPU 图像预处理开关, 缓解显存 OOM
- 推荐动作:
 - 是否精读: 若部署涉及 Kimi-K2.5 或大量 tokenizer worker, 值得精读并实测; 其余场景看结论即可——默认行为不变, 需要时设置 `SGLANG_FORCE_CPU_IMAGE_PREPROCESSING=1`。
 - 值得关注的设计决策: 一是 4 次提交的收敛路径 (从“尊重 image processor backend”到“预处理与全局 GPU 解码解耦”, 最后回到“一个通用 flag 简单控制”), 体现了在该粒度上选择最小实现; 二是 `gpu_image_decode` 与分派条件共用同一常量, 强制解码与预处理同设备, 避免 GPU→CPU 像素回拷; 三是开关在 import 期一次性求值, 值得与其他 env 的读取时机一致性做核对。
 - 建议后续补做: 仓库内单元测试 (覆盖 flag 未设 / 0 / 1、CUDA 有无、有图 / 无图分支)、开启前后的输出等价性校验、`--tokenizer-worker-num 16` 下的显存与吞吐回归数据, 并补一行用户文档说明该开关当前仅对 Kimi-K2.5 生效。

功能与动机

PR body 明确指出问题场景: “With many tokenizer workers (for example, `--tokenizer-worker-num 16`) sharing the first GPU, GPU image processing can cause out-of-memory errors.”; 同时说明为什么不在全局层面统一处理: “A global CPU preprocessing switch is difficult to apply consistently because models use different processing paths and hardcoded device choices.”。因此选择新增一个通用命名的开关 `SGLANG_FORCE_CPU_IMAGE_PREPROCESSING`, 但只在 Kimi-K2.5 上落地支持, 为后续模型逐步接入留出命名空间。

实现拆解

按「配置声明 → 处理器接线 → 设备一致性 → 验证配套」四步拆解:

1. 注册全局开关 (配置入口): `python/sglang/srt/envron.py` 的 `class Envs` 「Multimodal processing」段新增 `SGLANG_FORCE_CPU_IMAGE_PREPROCESSING = EnvBool(False)`, 注释写明 “Currently supported by the Kimi-K2.5 image processor only.”。默认 `False` 保证既有部署零变化, 复用 `EnvBool` 使 `0 / 1 / false / true` 都能正确解析 (PR 验证覆盖了这 5 种取值)。影响面: 为其他多模态模型预留了通用命名, 但当前

只有 Kimi 读取它。

2. 处理器侧接线 (import 期固化常量) : `python/sglang/srt/multimodal/processors/kimi_k25.py` 顶部新增 `from sglang.srt.environ import envs`, 并定义模块级常量 `_FORCE_CPU_IMAGE_PREPROCESSING = envs.SGLANG_FORCE_CPU_IMAGE_PREPROCESSING.get()`。选择导入时求值, 是为了让判定结果在进程生命周期内稳定; 代价是必须在启动服务器前设置环境变量, 运行中修改或 import 后 `monkeypatch os.environ` 都不会生效。
3. 改写 GPU / CPU 分派条件 (核心逻辑) : `KimiGPUProcessorWrapper.__call__` 中原判定 `if images and torch.cuda.is_available()`: 改为 `if images and not _FORCE_CPU_IMAGE_PREPROCESSING and torch.cuda.is_available()`: 不满足时落到既有的 `_cpu_call` (HF CPU 处理器路径)。开关只做“降级”而不新增实现, CPU 路径复用 `media_processor`, 并保留 `preserve_processor_input_ids` 与 `image_grid_thw` 键的既有处理, 因此请求自带 token id 的语义不变。
4. 同步解码设备 (避免跨设备搬运) : `KimiK2_5VLIImageProcessor.gpu_image_decode` 由硬编码 `True` 改为 `not _FORCE_CPU_IMAGE_PREPROCESSING`。原因: 若只关掉 GPU 预处理而保留 GPU 解码, 像素仍需从 GPU 拷回 CPU, 收益被抵消; 两者共用同一常量可保证“解码设备 = 预处理设备”。副作用是 JPEG 也放弃 `nvJPEG`、回到 CPU 解码。
5. 测试与验证配套 (缺口) : 本 PR 未新增或修改任何测试文件。PR body 与 review 描述的是仓库外临时脚本验证——20 项 `dispatch` 检查、24 项 `source-level routing` 检查, 覆盖 `flag` 未设置 / `0` / `1` / `false` / `true`、CUDA 可用与不可用、有图与空输入; 真实 Kimi 单测、GPU 集成、多 tokenizer worker 显存验证均标注为 `pending`。

关键文件:

- `python/sglang/srt/multimodal/processors/kimi_k25.py` (模块 多模态处理; 类别 `source`; 类型 `core-logic`; 符号 `_FORCE_CPU_IMAGE_PREPROCESSING`, `KimiGPUProcessorWrapper.call`, `KimiK2_5VLIImageProcessor.gpu_image_decode`) : Kimi-K2.5 多模态处理器的唯一落地点: 分派条件加入 CPU 强制分支, 并把解码设备与该开关绑定, 是本次行为的全部来源。
- `python/sglang/srt/envron.py` (模块 环境变量; 类别 `source`; 类型 `configuration`; 符号 `SGLANG_FORCE_CPU_IMAGE_PREPROCESSING`) : 新增全局布尔开关的声明处, 决定默认语义与取值解析方式, 是本次改动的配置入口。

关键符号: `KimiGPUProcessorWrapper.call`, `KimiK2_5VLIImageProcessor.gpu_image_decode`, `KimiGPUProcessorWrapper._gpu_call`, `KimiGPUProcessorWrapper._cpu_call`

关键源码片段

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

Kimi-K2.5 多模态处理器的唯一落地点: 分派条件加入 CPU 强制分支, 并把解码设备与该开关绑定, 是本次行为的全部来源。

```
# python/sglang/srt/multimodal/processors/kimi_k25.py
from sglang.srt.environ import envs

# 模块导入时只读一次: 进程启动后再改环境变量不会生效,
```

```

# 所以必须在 python -m sglang.launch_server 之前设置。
# 默认值 False 表示沿用既有 GPU 行为，不引入新的实现分支。
_FORCE_CPU_IMAGE_PREPROCESSING = envs.SGLANG_FORCE_CPU_IMAGE_
PREPROCESSING.get()

class KimiGPUProcessorWrapper:
    """包装 HF 的 Kimi 处理器：默认在 CUDA 上做 resize / pad / normalize / patchify。"""

    def __call__(self, text=None, images=None, **kwargs):
        # process_mm_data 会通过 kwargs["images"] 传入图像
        images = images or kwargs.pop("images", None)
        original_input_ids = kwargs.pop("sglang_original_input_ids", None)

        # 三个条件同时成立才走 GPU 快路径：
        # 1) 本次请求确实带图；
        # 2) 没有开启强制 CPU 预处理（本次新增的判定）；
        # 3) 当前进程可见 CUDA。
        # 任一不成立都落到 _cpu_call，即既有的 HF CPU 处理器，
        # 它按 preserve_processor_input_ids 保留请求自带的 token id。
        if images and not _FORCE_CPU_IMAGE_PREPROCESSING and torch.cuda.is_available():
            return self._gpu_call(text, images, original_input_ids)
        return self._cpu_call(text, images, original_input_ids, **kwargs)

class KimiK2_5VLIImageProcessor(KimiGridMMDataMixin, SGLangBaseProcessor):
    models = [KimiK25ForConditionalGeneration]

    # 原实现硬编码 True（JPEG 走 nvJPEG，其余走 PIL fallback）。
    # 与上面的分派条件共用同一常量，保证「解码设备 = 预处理设备」：
    # 若只关掉预处理而保留 GPU 解码，像素还得额外从 GPU 拷回 CPU，收益会被抵消。
    gpu_image_decode = not _FORCE_CPU_IMAGE_PREPROCESSING

    prefer_tokenized_input = True
    precompute_hash_before_cpu_transfer = True
    preserve_processor_input_ids = True
    auto_mm_processor_worker_num = 2
    auto_mm_io_worker_num = 16
    supports_mm_processor_concurrency = True

```

python/sglang/srt/envron.py

新增全局布尔开关的声明处，决定默认语义与取值解析方式，是本次改动的配置入口。

```

# python/sglang/srt/envron.py —— class Envs 中的「Multimodal processing」配置段
class Envs:
    # =====
    # Multimodal processing
    # =====
    SGLANG_VLM_CACHE_SIZE_MB = EnvInt(100)

```

```
SGLANG_IMAGE_MAX_PIXELS = EnvInt(16384 * 28 * 28)
SGLANG_RESIZE_RESAMPLE = EnvStr("")
SGLANG_MM_BUFFER_SIZE_MB = EnvInt(0)
SGLANG_MM_PRECOMPUTE_HASH = EnvBool(False)
SGLANG_VIT_ENABLE_CUDA_GRAPH = EnvBool(False)
# 使用全向量化的 ViT position-embedding 插值（无逐图 Python 循环、无 CPU<->GPU 同步）。
# 与旧实现逐位一致；设为 False 可回退到逐图循环。
SGLANG_VIT_ENABLE_VECTORIZED_POS_EMBED = EnvBool(True)
SGLANG_MM_SKIP_COMPUTE_HASH = EnvBool(False)
# 新增开关：目前仅 Kimi-K2.5 的 image processor 会读取它；
# 默认 False，保持「CUDA 可用则优先 GPU 预处理，否则回退 CPU」的既有语义。
# EnvBool 使 0 / 1 / false / true 都能被正确解析。
SGLANG_FORCE_CPU_IMAGE_PREPROCESSING = EnvBool(False)
```

评论区精华

本 PR 只有 2 条 review 记录：作者 ByronHsu 的自评（COMMENTED，含完整审查结论）与维护者 mickqian 的 APPROVED，另有 issue 评论“/tag-and-rerun-ci”触发 CI 重跑。关键交锋点集中在设计归属、默认开销与测试缺口三处：

“The change keeps ownership in the K2.5 processor: the same startup flag controls decoding and preprocessing, the existing HF fallback preserves request token IDs and normalizes the grid key, and the default GPU path remains intact. Kimi-K3 overrides both its dispatch and decode policy, so its behavior is unchanged.”——作者自评确认改动局限在 K2.5 处理器内部，K3 等覆盖了自身 dispatch / decode 策略的变体不受影响。

“The default path adds one boolean check per processor call, with no new tensor allocation, transfer, or synchronization.”——对默认路径性能影响给出的量化结论：只有一次布尔判断，无新增分配或同步。

“CUDA and processor calls were mocked; these checks do not validate pixel parity or GPU memory usage. I could not rerun the full unit suite or GPU image tests in this local environment. One non-blocking test-coverage suggestion is inline.”——作者自己点明验证边界：mock 只覆盖分派路由，不覆盖像素等价性与显存收益；并留了一条未在本 PR 落地的测试覆盖建议（该 inline 评论在提供的材料中不可见，具体内容不确定）。

mickqian 的 APPROVED 未附文字，说明在非详细审核下接受当前最小实现；也就是除作者外只有一名维护者确认。

- 设计归属：同一开关同时控制解码与预处理，避免设备混用 (design): 认可把开关留在 K2.5 处理器内部，并让 gpu_image_decode 与分派条件共用同一常量，规避 GPU 解码 + CPU 预处理的跨设备像素回拷；无阻塞性问题。
- 默认路径开销与跨模型兼容性 (performance): 默认路径无新增分配 / 传输 / 同步，其他模型行为不变；维护者 mickqian 以 APPROVED 接受。
- 测试覆盖缺口：仅有 mock 验证，缺像素等价与显存证据 (testing): 本 PR 未落地任何仓库内测试，属已知缺口；动机场景（16 个 tokenizer worker 的显存 OOM）与输出一致性仍未被端到端验证。

风险与影响

- 风险：

1. 缺少仓库内测试覆盖（主要风险）：python/sglang/srt/multimodal/processors/kimi_k25.py 的分派条件与 gpu_image_decode 都由模块级常量驱动，仓库中没有任何测试锁住这两个行为；后续若重构常量位置或改回硬编码，回归不会被 CI 捕获。reviewer 也明确说明 mock 校验 “do not validate pixel parity or GPU memory usage”，即开关开启后输出张量是否与 GPU 路径一致没有证据。
2. import 期读取导致不可动态切换：_FORCE_CPU_IMAGE_PREPROCESSING 在模块导入时求值一次，任何“先 import 再设环境变量”或进程内动态切换都无效；对测试注入也不友好（需要 patch 模块属性而非 os.environ）。
3. CPU 路径吞吐退化未量化：开启后丢失 nvJPEG，预处理回到 HF CPU 实现，而动机场景恰恰是 16 个 tokenizer worker 共享一卡，CPU 侧可能成为新瓶颈；PR 未给出吞吐 / 延迟数据。
4. 通用命名与单模型支持不一致：变量名 SGLANG_FORCE_CPU_IMAGE_PREPROCESSING 是全局语义，但只有 Kimi-K2.5 读取，其他多模态模型上设置它会静默无效，environ.py 里的一行注释是目前唯一的提示。
5. CI 状态非全绿：PR body 显示 PR Test (Base) 为 :no_entry_sign:、PR Test (Extra) 与 AMD ROCm 10 为 :x:；Kimi-K2.5 属 NVIDIA 场景，AMD 失败与本次改动的关系需确认，不能据此推断实现正确性，也不能据此判定实现有错。

- 影响：

- 用户影响：默认 False，绝大多数用户零感知，仅在每次 processor 调用多一次布尔判断；只有显式设置该变量并部署 Kimi-K2.5 的运维者会切到 CPU 预处理，预期换来更低 GPU 显存占用、更高 CPU 占用与潜在吞吐下降。文档未新增说明，用户只能通过 PR body 或 environ.py 注释发现该开关。
- 系统影响：改动只触及 Kimi-K2.5 多模态处理器单点，不涉及调度器、KV cache、注意力后端等共享路径，跨模型回归面很小；Kimi-K3 等覆盖自身 dispatch / decode 策略的变体行为不变（reviewer 已核对）。
- 团队影响：仓库中又多了一个“通用命名 + 单模型落地”的 env 开关，后续模型接入时需要决定是扩散实现还是补文档；同时把多 tokenizer worker 场景下 GPU 图像预处理的显存问题显性化，可能引出更系统的 CPU 预处理策略。
- 风险标记：缺少仓库内测试覆盖，import 期读取环境变量，CPU 路径吞吐退化未量化，通用命名仅 Kimi 支持，CI 未全绿

关联脉络

- PR #39278 [Fix][Qwen-VL] Normalize sentinel on artifact fast path: 同为多模态 processor 的快路径 / 回退路径改造，关注 CPU 与 GPU 分支的一致性，可作为同类改动的对照。
- PR #39347 Allow CUDA VMM feature transport with the Rust frontend: 同样是通过配置开关为多模态相关能力放行，并涉及服务端参数校验的调整。

- PR #38554 [Spec] Allow speculative workers to stage prefill shared reads: 采用相同的「默认关闭、显式开启」opt-in 特性开关模式，保持上游默认行为不变，可对照其测试配套方式。