PR #35700 完整报告

sgl-project/sglang

fix(multimodal): keep LLaVA image fetch off the CPU-preprocess timeout budget (flaky test_mixed_batch)

合并时间：2026-08-21 01:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35700

执行摘要

- 一句话：LLaVA 图像加载移出 CPU 超时预算，修复 flaky 500
- 推荐动作：建议精读 llava.py 中 _process_single_image、_fetch_remote_image_bytes、_preprocess_image_task 的拆分方式，这是异步超时预算与执行器池结合设计的典型案例；修复有量化证据但缺少固化测试，可关注后续是否补测，并在文档中补充 SGLANG_MM_LOAD_MAX_RETRIES 说明。

功能与动机

PR body 指出 test_mixed_batch 因 raw.githubusercontent.com 在 CI 网络拥塞下获取慢导致 HTTP 500；ProcessPoolExecutor 的 future 一旦运行无法取消，超时后 worker 仍占用池，12 个并发请求会在 4 线程客户端下打满池并级联失败，这同时是影响所有 LLaVA 家族模型的生产稳健性缺陷。其他 VLM 走 BaseMultimodalProcessor.load_mm_data 已分离 io 与 CPU 工作，因此只有 LLaVA 暴露此问题。

实现拆解

1. 拆分任务函数：原静态方法 _process_single_image_task 重命名为 _preprocess_image_task，去掉网络获取逻辑，只接收已获取的图片输入（远程压缩字节、本地路径或内联字节），按 pad / anyres 分支完成解码与预处理。改动集中在 python/sglang/srt/multimodal/processors/llava.py。
2. 新增 io 池获取函数：_fetch_remote_image_bytes 通过 self.io_executor 调用 get_image_bytes，对 requests.exceptions.Timeout 与 ConnectionError 做最多 SGLANG_MM_LOAD_MAX_RETRIES（默认 2）次重试，指数退避从 0.5 秒翻倍；每次尝试内部由 download_remote_media 的 REQUEST_TIMEOUT 兜底，总时长有界。
3. 重新编排 _process_single_image：先统一提取 URL 并预计算 image_hash；远程图片先经 io 池取回压缩字节再交给 CPU 池，本地 / 内联输入直接进 CPU 池；asyncio.wait_for 现在只约束 CPU 段，网络等待不再挤占该预算。
4. 配套与验证：无新增配置键（只有环境变量），未新增测试文件；PR 通过故障注入（1×H200、p=0.12 慢获取、REQUEST_TIMEOUT=3、SGLANG_CPU_WORKERS=4、禁用客户端重试）各跑 100 次对比，每请求 500 率从 7.95%（159/2001）降为 0（0/1200），test_mixed_batch 单次失败率 67%→0；真实 CI 测试 test_vision_openai_server_a.py -k TestLlavaServer 6 passed。

关键文件：

- python/sglang/srt/multimodal/processors/llava.py（模块 多模态；类别 source；类型 core-logic；符号 _process_single_image_task, _preprocess_image_task, _fetch_remote_image_bytes, _process_single_image）：唯一变更文件，将 LLaVA 图片的网络获取从 CPU 预处理超时预算中拆出，引入 io 线程池获取与指数退避重试，是 flaky 修复与生产稳健性改进的核心。

关键符号：_fetch_remote_image_bytes, _preprocess_image_task, _process_single_image

关键源码片段

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

唯一变更文件，将 LLaVA 图片的网络获取从 CPU 预处理超时预算中拆出，引入 io 线程池获取与指数退避重试，是 flaky 修复与生产稳健性改进的核心。

```
# python/sglang/srt/multimodal/processors/llava.py（head 版本整理）

import asyncio
import os

import requests

from sglang.srt.utils import ImageData, get_image_bytes, load_image


class LlavaImageProcessor(BaseMultimodalProcessor):
    @staticmethod
    def _preprocess_image_task(image_input, image_hash, image_aspect_ratio,
                               image_grid_pinpoints, processor=None):
        # CPU 侧解码 + 预处理。此时 image_input 是已获取的压缩字节（远程）
        # 或本地 / 内联输入，不会发生网络等待，适合放在 ProcessPool 并行执行。
        image_processor = processor.image_processor
        try:
            image, image_size = load_image(image_input, False)
            if image_size is not None:
                # 视频或多帧场景：逐帧预处理后堆叠
                pixel_values = image_processor(image)["pixel_values"]
                for i in range(len(pixel_values)):
                    pixel_values[i] = ensure_numpy(pixel_values[i]).astype(np.float16)
                return np.stack(pixel_values, axis=0), image_hash, image_size
            # 普通图片：按 aspect_ratio 分支处理
            if image_aspect_ratio == "pad":
                image = expand2square(
                    image, tuple(int(x * 255) for x in image_processor.image_mean)
                )
                pixel_values = image_processor(image.convert("RGB"))["pixel_values"][0]
            elif image_aspect_ratio == "anyres" or "anyres_max" in (image_aspect_ratio or ""):
                pixel_values = process_anyres_image(
```

```
                image, image_processor, image_grid_pinpoints
            )
        else:
            pixel_values = image_processor(image)["pixel_values"][0]
        pixel_values = ensure_numpy(pixel_values)
        if isinstance(pixel_values, np.ndarray):
            pixel_values = pixel_values.astype(np.float16)
        return pixel_values, image_hash, image.size
    except Exception:
        logger.error("Exception in TokenizerManager:\n" + get_exception_traceback())

async def _fetch_remote_image_bytes(self, url):
    # 网络获取移到 io 线程池，避免占用 CPU 池；只重试瞬态网络错误。
    # 每次尝试由 download_remote_media 内部的 REQUEST_TIMEOUT 兜底。
    loop = asyncio.get_running_loop()
    max_retries = max(0, int(os.environ.get("SGLANG_MM_LOAD_MAX_RETRIES", "2")))
    delay = 0.5
    for attempt in range(max_retries + 1):
        try:
            return await loop.run_in_executor(self.io_executor, get_image_bytes, url)
        except (requests.exceptions.Timeout, requests.exceptions.ConnectionError):
            if attempt >= max_retries:
                raise
            await asyncio.sleep(delay)
            delay *= 2

async def _process_single_image(self, image_data, aspect_ratio, grid_pinpoints):
    # 统一提取 URL 并预计算哈希；远程图片先由 io 池拿压缩字节，
    # CPU 池只做解码与预处理，asyncio.wait_for 不再包含网络等待。
    url = image_data.url if isinstance(image_data, ImageData) else image_data
    image_hash = hash(url)
    if isinstance(image_data, ImageData) and image_data.url:
        image_input = await self._fetch_remote_image_bytes(url)
    else:
        image_input = image_data # 本地路径或内联字节
    loop = asyncio.get_running_loop()
    fut = loop.run_in_executor(
        self.cpu_executor,
        LlavaImageProcessor._preprocess_image_task,
        image_input, image_hash, aspect_ratio, grid_pinpoints, self._processor,
    )
    timeout = int(os.environ.get("REQUEST_TIMEOUT", "10"))
    return await asyncio.wait_for(fut, timeout=timeout)
```

评论区精华

第二轮评审（记录在 commit 089be56b）指出，先前版本在 io 池加载图片后把懒加载 PIL 对象传入 ProcessPool，pickle 会迫使解码发生在 tokenizer 进程，并把原始像素跨进程传输（约 100 倍压缩大小），序列化与内存开销巨大。结论是 io 池只取压缩字节、CPU 池负责解码

，跨进程数据量最小化。此外 CI 反复触发 /rerun-test test/registered/vlm/test_vision_openai_server_a.py，多次验证均通过，证明 flaky 修复在真实 CI 网络下稳定。

- io 池与 CPU 池的分工与 pickle 序列化问题 (design): 改为 io 池只取压缩字节，CPU 池负责解码与预处理，跨进程数据量最小化。

风险与影响

- 风险：
  1. 异常传播行为变化：此前网络异常会被 _process_single_image_task 的 try/except 捕获并记录日志后返回 None；现在 _fetch_remote_image_bytes 对非 Timeout/ConnectionError 异常（如 HTTP 404）会直接上抛，最终请求以 500 结束，从“静默失败”变为“显式失败”，需确认上层对 None 的容错逻辑不会引入空指针类问题。
  2. 重试导致最坏延迟上升：默认 SGLANG_MM_LOAD_MAX_RETRIES=2，每次请求最多 3 次网络尝试，最坏情况下单请求图片加载时长约为 3 × REQUEST_TIMEOUT + 退避等待，对端到端延迟敏感场景需评估。
  3. 缺少固化测试：故障注入实验是 PR body 中的手工基准，未沉淀为仓库内单元 / 集成测试，后续重构执行器或超时逻辑时有回归风险。
  4. 影响范围：变更只涉及 LLaVA 家族处理器（LlavaLlama / LlavaVid / LlavaQwen / LlavaMistral）的图片加载路径，其他 VLM 不受影响。 - 影响：对用户：修复 LLaVA 图片源瞬态缓慢时的 HTTP 500，生产稳定性提升，重试机制增强对瞬态网络错误的容错。对系统：io/cpu 分工更明确，ProcessPool 不再被慢网络请求占满，消除级联 500；跨进程只传压缩字节，传输数据量约减小 100 倍。对团队：为多模态处理器提供与 BaseMultimodalProcessor 一致的 io/cpu 分离模式，新环境变量 SGLANG_MM_LOAD_MAX_RETRIES 值得纳入文档。 - 风险标记：核心路径变更，异常传播行为变化，重试最坏延迟上升，缺少固化测试

关联脉络

- 暂无明显关联 PR