PR #35342 完整报告

sgl-project/sglang

[VLM] Route every multimodal processor through the worker pool's call site

合并时间：2026-08-27 10:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35342

执行摘要

- 一句话：多模态处理器统一改走 worker 池异步调用点
- 推荐动作：值得精读，重点看三处：新增测试文件的 AST 扫描设计（用静态分析锁定调用点不变量）、PR body 对身份变换的论证（await 不挂起则行为不变）、以及开启并发前的性能权衡方法论（GIL 释放、prefill 批碎裂）。它是典型的大范围机械重构 + 防回退测试的样板，适合作为后续批量迁移的参考。

功能与动机

PR body 指出 process_and_combine_mm_data 是 multimodal processor worker pool 实际运行的函数，但 37 个调用点中有 33 个直接调用它，导致这些模型即使配置 --mm-processor-worker-num > 1 也会在启动时构建线程池和 deepcopy 处理器克隆，却把每个请求直接绕过池子——这是静默失败，模型始终以单 worker 速度服务。PaddleOCR-VL 从第一天起就是一个例子：接入并发（#35318）后，H200 上 32 路并发吞吐从 6.7 req/s 提升到 10.9 req/s，纯粹靠 87 ms/page 的预处理与其他请求重叠。

实现拆解

实现分为 4 步：

1. 盘点调用点：用 AST 扫描 python/sglang/srt/multimodal/processors 目录，确认 37 个调用点，其中 33 个直连同步版，只有 qwen_vl、kimi_k25、kimi_k3、muse_glimmer 已在正确的异步入口。
2. 机械迁移：对 33 个调用点逐一执行 await + _async 改名，覆盖 clip.py、cohere2_vision.py、deepseek_ocr.py、deepseek_vl_v2.py、dots_vlm.py、ernie45_vl.py、gemma3.py、gemma3n.py、gemma4.py、glm4v.py、glm_image.py 等文件；所有调用点均位于 async def 内，因此 await 合法。Rebase 时处理了 transformers_auto.py 新增调用点和 pixtral.py、qwen_audio.py 冲突。
3. 新增审计测试：test/registered/unit/multimodal/test_processor_async_call_sites.py 用 AST 扫描整个 multimodal 目录，断言所有 process_and_combine_mm_data* 调用必须以 _async 结尾；同时验证每个调用点都位于 async def 内，并断言调用点数量大于 20 防止扫描静默失效。测试注册为 CPU CI（est_time=3）。
4. 验证：作者用脚本机械核对 33 个旧调用点清零、每个改动行仅差 await 和 _async 后缀，且 isort、black、ruff 全部通过。由于是身份变换，本 PR 未做 benchmark。

关键文件：

- test/registered/unit/multimodal/test_processor_async_call_sites.py（模块 审计测试；类别 test；类型 test-coverage；符号 _enclosing_function, _call_sites, test_no_processor_bypasses_the_worker_pool, test_every_call_site_can_await）：新增 AST 审计测试，锁定所有 multi-modal processor 必须走异步调用点，是本 PR 的核心保障；rebase 中发现 transformers_auto.py 新调用点就归功于它。
- python/sglang/srt/multimodal/processors/clip.py（模块 多模态处理；类别 source；类型 core-logic；符号 process_mm_data_async）：CLIP 处理器是 33 个迁移文件之一，展示 sync 调用改为 await async 变体的标准形态，适合作为代表性样例。
- python/sglang/srt/multimodal/processors/deepseek_ocr.py（模块 多模态处理；类别 source；类型 core-logic；符号 process_mm_data_async）：DeepseekOCRProcessor 是 deepseek 标签关联的处理器代表，同样从 sync 切换到 async 调用点，体现本次迁移覆盖 deepseek 系模型。

关键符号：process_and_combine_mm_data, process_and_combine_mm_data_async, process_mm_data_async, _enclosing_function, _call_sites, test_no_processor_bypasses_the_worker_pool, test_every_call_site_can_await, test_the_scan_actually_finds_call_sites

关键源码片段

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

CLIP 处理器是 33 个迁移文件之一，展示 sync 调用改为 await async 变体的标准形态，适合作为代表性样例。

```
# 文件：python/sglang/srt/multimodal/processors/clip.py
# ClipImageProcessor.process_mm_data_async 是这次迁移的典型形态。

async def process_mm_data_async(
    self, image_data: List[Union[str, bytes]], input_text, *args, **kwargs
):
    # 先加载并解析图像与文本数据，得到统一的 base_output
    base_output = await self.load_mm_data(
        prompt=input_text,
        multimodal_tokens=self.mm_tokens,
        image_data=image_data,
    )

    # 关键改动：原来直接调用同步函数 process_and_combine_mm_data，
    # 现在改为 await process_and_combine_mm_data_async(...)。
    # 无 executor 时该协程直接委托给同步实现，行为完全不变；
    # 有 executor 时才经 run_in_executor 进入线程池，实现并发。
    mm_items, input_ids, _ = await self.process_and_combine_mm_data_async(
        base_output, self.mm_tokens
    )

    return MultimodalProcessorOutput(
```

```
        mm_items=mm_items,
        input_ids=input_ids.tolist(),
    )
```

评论区精华

PR 的 review 评论为空，核心讨论集中在 PR body 和 issue 评论中：

- 作者在 Notes for reviewers 中明确把"开启并发"排除在本 PR 之外，并给出权衡：ThreadPoolExecutor 只在预处理释放 GIL 时才有效（PIL、numpy、torch 的 C 代码可以，纯 Python 预处理不行）；更多 worker 会分散请求到达，导致 GPU prefill 批处理碎片化，qwen_vl 选 2 个 worker、PaddleOCR-VL 上 8 个 worker 反而比 4 个慢。
- rebase 说明中提到 transformers_auto.py 在 main 上新增的调用点正是被本 PR 自己的审计测试捕获，说明审计测试在实际开发中起了作用；pixtral.py 和 qwen_audio.py 的冲突以 main 版本为准。
- issue 评论中作者解释两次 CI 失败均为 runner 环境问题：XPU 镜像缺 libsycl.so.8，NPU 脚本无测试输出，与本 PR 无关，且这两个 job 已在 #35238 中排除出 fast-fail 级联。
- CI 失败是否由本 PR 引入 (other): 这些 job 已在 #35238 中排除出 fast-fail 级联，失败与本 PR 无关；最终 nightly CI 未发现由改动引入的失败。
- 是否在本 PR 内直接开启并发 (design): 本 PR 只统一调用点，不开启任何模型的并发；后续每个模型需单独测量后再接入。
- Rebase 冲突与审计测试的自证 (other): 审计测试在 rebase 中实际发现了漏网调用点，证明 AST 审计的必要性；冲突解决后仍是身份变换。

风险与影响

- 风险：
  - 行为不变性的前提：identity transform 依赖 process_and_combine_mm_data_async 在无 executor 时直接委托同步函数且不挂起；若未来该函数增加挂起点（如引入 await 其他协程），请求交错行为就会改变。
  - AST 扫描维护成本：审计测试依赖 _MULTIMODAL_ROOT 路径和 _EXEMPT 豁免名单（base_processor.py）；新增处理器目录或改名后需要同步维护，否则可能漏检或误报。
  - 并发配置风险：本 PR 不开启并发，但为后续开启铺路；开启时若不逐个模型测量，可能出现吞吐反而下降（PaddleOCR-VL 8 workers 慢于 4 workers 的先例）。
  - 覆盖缺口：审计只锁定了调用点命名，不验证并发路径的正确性，也没有端到端回归测试；未来 executor 逻辑出错时可能不会被本测试发现。
- 影响：
  - 用户侧：行为逐位不变，现有部署零风险；未来模型只需三行改动（加能力开关 + 确认调用点在异步入口）即可接入并发预处理。
  - 系统侧：37 个调用点全部统一到异步入口，消除了"启动时构建线程池但请求全部绕过"的静默单 worker 模式；PaddleOCR-VL 等重预处理模型可直接受益。
  - 团队侧：新增的 AST 审计测试成为 permanent guard，main 上新增的 transformers_auto.py 调用点正是被它发现的，验证了测试价值。

- 风险标记：依赖 async 委托语义，AST 扫描路径维护，并发开关需逐个验证，rebase 后新增调用点易遗漏，无端到端并发回归测试

关联脉络

- PR #35318 Enable PaddleOCR-VL mm processor concurrency: PR body 引用该 PR 作为 PaddleOCR-VL 接入并发的实例：32 路并发吞吐从 6.7 提升到 10.9 req/s，是本次统一调用点的动机来源。
- PR #35238 Adjust CI fast-fail cascade exclusions: issue 评论中提到该 PR 将 XPU/NPU 等环境敏感 job 排除出 fast-fail 级联，解释了 CI 失败与本 PR 无关。