PR #24751 完整报告

sgl-project/sglang

fix(mm): make multimodal data loading non-blocking to prevent health check stalls

合并时间：2026-05-22 10:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24751

执行摘要

- 一句话：多模态数据加载改为非阻塞，修复健康检查卡死
- 推荐动作：建议精读本 PR，特别是 base_processor.py 中的异步转换模式，可作为类似 event loop 阻塞修复的参考。注意：未来新增子处理器时，需确保调用 await self.load_mm_data()。

功能与动机

修复 issue #24738：当多模态请求包含慢速视频 URL（如不可达地址或大文件）时，load_mm_data 中的同步 future.result() 会阻塞事件循环，导致服务器无法响应健康检查请求，进而被编排器（如 K8s）误重启。

实现拆解

1. 在 base_processor.py 中添加 import asyncio，并将 load_mm_data、fast_load_mm_data、legacy_load_mm_data 的 def 改为 async def。
2. 将内部的 future.result() 调用替换为 await asyncio.wrap_future(future)，使得等待 I/O 时让出事件循环。
3. 遍历所有继承自 BaseMultimodalProcessor 的子处理器（如 internvl.py、minicpm.py、clip.py 等），将所有调用 self.load_mm_data(...) 的地方加上 await，因为这些调用已处于 async def process_mm_data_async 上下文中。
4. 针对后合并的 MiniCPM-V 4.6 处理器，在第二个 commit 中补充将其 patch 为异步调用。
5. 没有新增测试配套，但作者手动验证了 15 个并发坏 URL 请求 + health check 场景，确认修复有效。

关键文件：

- python/sglang/srt/multimodal/processors/base_processor.py（模块 多模态处理器；类别 source；类型 core-logic；符号 load_mm_data, fast_load_mm_data, legacy_load_mm_data）：核心文件，将三个数据加载函数从同步改为异步，使用 asyncio.wrap_future 替换 future.result()，并添加 import asyncio。
- python/sglang/srt/multimodal/processors/internvl.py（模块 多模态处理器；类别 source；类型 core-logic）：需要更新子处理器中调用 load_mm_data 的地方，添加 await 关键字。

- python/sglang/srt/multimodal/processors/minicpm.py（模块 多模态处理器；类别 source；类型 core-logic）：需要更新子处理器中调用 load_mm_data 的地方，添加 await 关键字。
- python/sglang/srt/multimodal/processors/clip.py（模块 多模态处理器；类别 source；类型 core-logic）：需要更新子处理器中调用 load_mm_data 的地方，添加 await 关键字。
- python/sglang/srt/multimodal/processors/deepseek_ocr.py（模块 多模态处理器；类别 source；类型 core-logic）：需要更新子处理器中调用 load_mm_data 的地方，添加 await 关键字。

关键符号：load_mm_data, fast_load_mm_data, legacy_load_mm_data

关键源码片段

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

核心文件，将三个数据加载函数从同步改为异步，使用 asyncio.wrap_future 替换 future.result()，并添加 import asyncio。

```
# base_processor.py 核心变更：将同步数据加载转换为非阻塞异步

import asyncio # 新增导入，用于 asyncio.wrap_future

class BaseMultimodalProcessor(...):
    # 原有同步方法改为 async
    async def load_mm_data(self, prompt, ...):
        # ... 校验与预处理逻辑 ...
        if ...:
            # 原为 return self.legacy_load_mm_data(...) 同步阻塞
            return await self.legacy_load_mm_data(...)
        # 原为 return self.fast_load_mm_data(...)
        return await self.fast_load_mm_data(...)

    async def fast_load_mm_data(self, ...):
        # ... 提交任务到线程池 ...
        for modality, idx, future in futures:
            try:
                # 替换 future.result() 为 await asyncio.wrap_future(future)
                result = await asyncio.wrap_future(future)
            except Exception as e:
                logger.exception(...)
        # ...

    async def legacy_load_mm_data(self, ...):
        # ... 模态匹配循环 ...
        for text_part in text_parts:
            if ...:
                # 同样替换
                result = await asyncio.wrap_future(next(futures_iter))
                # ...
```

评论区精华

AgainstEntropy 进行了 *e2e* 测试验证：15 个并发坏 URL 请求后 health check 仍能正常响应（0.999987s），而 main 分支超时（15.002s）。同时讨论了保持本 PR 范围干净，将其他模型覆盖作为后续工作。作者 abinggo 手动核查了 processors 目录下所有文件，确认 7 个未修改的文件中 2 个不调用 load_mm_data，其余通过父类继承已覆盖。

- Follow-up for other models (design): 作者同意保持范围干净，后续单独 PR 处理其他模型。
- Impact on unaffected processors (testing): 所有现有 processor 都在覆盖范围内。

风险与影响

- 风险：改动集中在多模态数据加载路径，不涉及推理循环、调度器或 CUDA kernel。主要风险是某个子处理器遗漏 await 导致运行时错误。作者已手动逐文件检查确保覆盖。但缺少自动化测试，未来新增处理器时可能遗漏异步调用。
- 影响：对用户：修复了多模态场景下 health check 超时问题，提升服务稳定性。对系统：async 调用让出事件循环，不影响其他请求的吞吐。对团队：变更范围清晰，易于审查。
- 风险标记：缺少自动化测试，核心路径变更

关联脉络

- 暂无明显关联 PR