PR #36983 完整报告

sgl-project/sglang

fix(vlm): recover multimodal decode and processor failures

合并时间：2026-08-30 20:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36983

执行摘要

- 一句话：修复 VLM 解码错误分类与预处理池故障恢复
- 推荐动作：值得精读。该 PR 展示了如何优雅处理多进程池故障：将阻塞提交移出事件循环、用锁保证并发替换幂等、用后台线程清理旧池，以及用 asyncio.shield 配合 gather(return_exceptions=True) 做取消安全的资源清理。这些模式可复用于其他依赖进程池且需要异步超时的场景。建议关注池创建失败和取消延迟两个边界，后续可考虑补充对应测试。

功能与动机

PR body 指出 Bad media and worker failures are request-scoped conditions，应当产生清晰的 VLM 请求错误并让处理器池继续服务后续请求。作者在评论中给出真实服务 A/B 验证：origin/main 对损坏 PNG 返回 HTTP 500，而当前 head 返回 HTTP 400 并给出明确报错，随后正常请求返回 200，/health 保持 200，说明此前损坏媒体会污染整个预处理服务。

实现拆解

1. 解码边界统一：在 python/sglang/srt/utils/common.py 新增 _fully_load_pil_image，将 PIL 懒加载解码的 OSError 统一转为 ValueError，并在 load_image 入口直接调用，同时 MultimodalPreprocessCache 的 _snapshot_pil 也做同样处理，确保损坏图片在进入处理器前就完成分类。
2. 进程池恢复框架：在 python/sglang/srt/multimodal/processors/base_processor.py 将 cpu_executor 的创建抽到 _create_cpu_executor，并新增 _replace_broken_cpu_executor 与 _shutdown_broken_cpu_executor；用 threading.Lock 保证并发请求下只会替换一次池，旧池的 shutdown 放到守护线程执行，避免阻塞事件循环。
3. LLaVA 任务提交重写：在 python/sglang/srt/multimodal/processors/llava.py 的 _process_single_image 中，将 ProcessPoolExecutor.submit 放进 asyncio.to_thread 以避免 worker 退出后 submit 阻塞事件循环，并为提交和等待结果分别设置超时；捕获 BrokenProcessPool/TimeoutError 后触发池替换。同时 _preprocess_image_task 对 CLIENT_MEDIA_EXCEPTIONS 抛 ValueError，其余异常记录 traceback 后重新抛出，不再静默吞错。
4. MOSS 视频归一化清理：在 python/sglang/srt/multimodal/processors/moss_vl.py 中，_normalize_video_inputs_async 改用 asyncio.gather(return_exceptions=True) +

asyncio.shield 等待所有 worker 完成或取消后再统一清理临时文件，新增 _remove_temp_video_paths 复用清理逻辑；_write_video_bytes_to_tempfile 写入失败时立即清理已创建文件。

5. 测试配套：新增 test_llava_processor_pool.py 覆盖坏池替换、阻塞提交超时、真实 worker 退出恢复、并发替换仅一次等场景；扩展 test_moss_vl_processor.py 覆盖归一化失败和取消时的临时文件清理；另在 test_llava.py、test_base_processor_bad_input.py、test_media_artifact_processor.py、test_kimi_k3_encoder_mode.py 中补充懒解码失败、OSError 源等断言。

关键文件：

- python/sglang/srt/multimodal/processors/base_processor.py（模块 处理器基类；类别 source；类型 core-logic；符号 _create_cpu_executor, _replace_broken_cpu_executor, _shutdown_broken_cpu_executor）：核心改动文件：抽出池创建、新增池替换与后台清理逻辑，加锁保证并发替换幂等，是整个故障恢复机制的基础。
- python/sglang/srt/multimodal/processors/llava.py（模块 图像处理；类别 source；类型 core-logic；符号 _process_single_image, _preprocess_image_task）：LLaVA 预处理任务提交路径重写：将 submit 移出事件循环、绑定超时，并在 BrokenProcessPool/Timeout 时触发池替换；同时保留并重新抛出预处理异常。
- python/sglang/srt/multimodal/processors/moss_vl.py（模块 视频处理；类别 source；类型 core-logic；符号 _normalize_video_inputs_async, _remove_temp_video_paths, _write_video_bytes_to_tempfile）：MOSS 视频归一化临时文件清理重构：异常与取消场景下统一清理，避免临时文件泄漏。
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 core-logic；符号 _fully_load_pil_image, _load_image, load_image）：解码边界分类：强制所有 PIL 图像 eager load，把 OSError 转为 ValueError，确保损坏媒体在通用入口即被识别。
- python/sglang/srt/multimodal/cache/identity.py（模块 缓存标识；类别 source；类型 core-logic；符号 _snapshot_pil）：媒体快照也会触发懒加载，需要同样分类解码错误，否则缓存路径仍会漏出 OSError。
- test/registered/unit/models/test_llava_processor_pool.py（模块 池恢复测试；类别 test；类型 test-coverage；符号 _BrokenExecutor, _BlockingExecutor, _exit_worker_process, test_llava_replaces_broken_pool_without_replaying_request）：新增测试文件，覆盖坏池替换、阻塞提交超时、真实 worker 退出恢复、并发替换幂等，是池恢复逻辑的完整验证。
- test/registered/unit/models/test_moss_vl_processor.py（模块 视频清理测试；类别 test；类型 test-coverage；符号 test_video_normalization_cleans_sibling_temp_file_on_failure, test_video_normalization_waits_for_worker_cleanup_when_cancelled）：新增 MOSS 视频归一化失败和取消场景的临时文件清理验证。
- test/registered/unit/models/test_llava.py（模块 LLaVA 测试；类别 test；类型 test-coverage；符号 TestLlavaImageProcessor, test_preprocess_reports_invalid_media_as_client_error）：补充 LLaVA 预处理对无效媒体抛 ValueError 且保留 cause 的测试。

- test/registered/unit/multimodal/test_base_processor_bad_input.py（模块 坏输入测试；类别 test；类型 test-coverage；符号 test_lazy_pil_decode_failure, test_image_source_os_error）：覆盖懒解码失败和 get_image_bytes 抛 OSError 的通用分类行为。

关键符号：_create_cpu_executor, _replace_broken_cpu_executor, _shutdown_broken_cpu_executor, _process_single_image, _preprocess_image_task, _normalize_video_inputs_async, _remove_temp_video_paths, _write_video_bytes_to_tempfile, _fully_load_pil_image, _snapshot_pil

关键源码片段

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

核心改动文件：抽出池创建、新增池替换与后台清理逻辑，加锁保证并发替换幂等，是整个故障恢复机制的基础。

```
# 创建 CPU 预处理进程池，根据 feature 传输方式选择 spawn / fork
# 单独抽出便于测试和故障替换复用
    def _create_cpu_executor(self) -> concurrent.futures.ProcessPoolExecutor:
        start_method = "spawn" if self.mm_feature_transport == "cuda_vmm" else "fork"
        return concurrent.futures.ProcessPoolExecutor(
            mp_context=mp.get_context(start_method),
            max_workers=int(os.environ.get("SGLANG_CPU_WORKERS", os.cpu_count())),
        )

    # 替换损坏的预处理池：用锁保证并发请求下只替换一次，
    # 并将旧池的 shutdown 放到后台线程，避免阻塞事件循环
    def _replace_broken_cpu_executor(
        self, failed_executor: concurrent.futures.ProcessPoolExecutor
    ) -> None:
        with self._cpu_executor_lock:
            if self.cpu_executor is not failed_executor:
                return # 已有其他请求完成替换，直接忽略
            self.cpu_executor = self._create_cpu_executor()
        logger.warning("Replaced a broken multimodal CPU preprocess pool")
        threading.Thread(
            target=self._shutdown_broken_cpu_executor,
            args=(failed_executor,),
            name="sglang-mm-cpu-pool-cleanup",
            daemon=True,
        ).start()

    @staticmethod
    def _shutdown_broken_cpu_executor(
        failed_executor: concurrent.futures.ProcessPoolExecutor,
    ) -> None:
        try:
            failed_executor.shutdown(wait=False, cancel_futures=True)
```

```
        except Exception:
            logger.warning(
                "Failed to shut down a broken multimodal CPU preprocess pool",
                exc_info=True,
            )
```

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

LLaVA 预处理任务提交路径重写：将 submit 移出事件循环、绑定超时，并在 BrokenProcessPool/Timeout 时触发池替换；同时保留并重新抛出预处理异常。

```
    # 处理单张图像：远程图片先拉字节，其余直接进 CPU 池
    # 关键：ProcessPoolExecutor.submit() 在 worker 退出后可能阻塞，
    # 因此把提交放到 to_thread，确保超时仍能替换坏池
    async def _process_single_image(self, image_data, aspect_ratio, grid_pinpoints):
        url = image_data.url if isinstance(image_data, ImageData) else image_data
        image_hash = hash(url) if isinstance(url, (str, bytes)) else None

        image_input = url
        if isinstance(url, str) and url.startswith(("http://", "https://")):
            image_input = await self._fetch_remote_image_bytes(url)

        if self.cpu_executor is not None:
            loop = asyncio.get_running_loop()
            executor = self.cpu_executor
            timeout = int(os.environ.get("REQUEST_TIMEOUT", "10"))
            deadline = loop.time() + timeout
            try:
                process_future = await asyncio.wait_for(
                    asyncio.to_thread(
                        executor.submit,
                        LlavaImageProcessor._preprocess_image_task,
                        image_input,
                        image_hash,
                        aspect_ratio,
                        grid_pinpoints,
                        self._processor,
                    ),
                    timeout=timeout,
                )
                remaining = max(0.0, deadline - loop.time())
                return await asyncio.wait_for(
                    asyncio.wrap_future(process_future), timeout=remaining
                )
            except (BrokenProcessPool, asyncio.TimeoutError):
                self._replace_broken_cpu_executor(executor)
                raise
        else:
            return LlavaImageProcessor._preprocess_image_task(
                image_input, image_hash, aspect_ratio, grid_pinpoints, self._processor
```

)

评论区精华

唯一一条评论来自作者 mickqian 的 A/B 测试说明：在 NVIDIA H200 上使用 Qwen3-VL-2B-Instruct 真实权重，origin/main 对 PNG 头可打开但像素懒解码失败的请求返回 HTTP 500；当前 head 返回 HTTP 400 并报 'Could not decode image: broken data stream'，且紧接着的正常图片请求返回 200，/health 保持 200。这直接验证了错误分类在解码边界生效且只影响当前请求。没有其他 review 讨论或反对意见。

- 真实服务 A/B 验证错误分类行为 (testing): 确认 malformed-media 故障在解码边界被分类，且请求本地化，不影响后续服务和健康检查。

风险与影响

- 风险： 1) 进程池替换逻辑涉及全局状态 cpu_executor，虽然加锁保证只替换一次，但若 _create_cpu_executor 本身抛异常，cpu_executor 仍指向坏池，后续请求会持续失败；当前代码未捕获创建失败场景。 2) LLaVA 的 asyncio.to_thread 提交在超时后返回，但底层线程可能仍在阻塞等待 submit，若池未真正损坏会造成线程泄漏；测试中的 _BlockingExecutor 用 5 秒超时规避了无限阻塞。 3) MOSS 的 _normalize_video_inputs_async 在取消时用 await gather_task 等待所有 worker 完成，若某个 worker 卡死会导致取消延迟；测试中通过 finish.set() 控制释放，真实场景可能更慢。 4) 错误类型从通用 500 转为 400 会改变客户端可见行为，依赖 HTTP 状态码做容错的客户端可能需要调整。 5) _fully_load_pil_image 对已有 PIL Image 路径也强制 load()，可能引入额外的内存拷贝和 CPU 开销，对超大图请求有潜在延迟影响。
  - 影响：影响所有使用 LLaVA 系、MOSS-VL 处理器以及 load_image 通用路径的多模态请求。修复后，损坏媒体将返回明确的 400 错误并保留异常链；预处理进程池遇到 worker 崩溃后可自动恢复，避免服务整体不可用。对运维而言，错误可观测性提升，故障不再表现为神秘 500。测试新增了约 170 行覆盖，注册到 CPU CI 套件，持续保障回归。
  - 风险标记：核心路径变更，进程池重建并发，异步取消清理，HTTP 状态码行为变更，池创建失败未覆盖

关联脉络

- 暂无明显关联 PR