PR #51120 完整报告

vllm-project/vllm

[Bugfix][Frontend] Return 400 for invalid PyNvVideoCodec video input

合并时间：2026-08-12 18:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51120

执行摘要

- 一句话：修复 PyNvVideoCodec 坏视频输入误报 HTTP 500，改为返回 400
- 推荐动作：值得精读。这是一个 " 跨库异常边界翻译 " 的教科书式案例：在不改动 API 层的前提下把厂商原生异常归一化为协议错误，并借助动态类型探测抵抗版本漂移。重点看 _pynvvideocodec_exception_types 的实现、两处 try/except 的归一化边界取舍，以及 review 中把 mock 单测升级为真实硬件集成测试的过程——对做多模态后端或 API 错误语义的工程师有直接借鉴价值。

功能与动机

用户提供的畸形视频在多模态请求预处理时到达 PyNvVideoCodec.SimpleDecoder，原生 PyNvVCException 此前逃逸出媒体边界，被 API 通用异常处理器归类为 InternalServerError——同一请求在 NIM 公共端点与内部 vLLM 端点均返回 HTTP 500，堆栈完全位于 vllm/multimodal/video.py / PyNvVideoCodec 且早于模型推理。这是客户端输入错误而非服务器故障，应当返回 HTTP 400 而不是 500，否则客户端无法区分输入错误与服务故障，监控也会被用户输入污染。

实现拆解

1. 新增异常类型探测辅助函数_pynvvideocodec_exception_types（vllm/multimodal/video.py）：动态扫描 PyNvVideoCodec 模块导出符号，收集所有名字以 PyNvVCException 开头、是类型且继承自 Exception 的异常类并返回 tuple。不硬编码厂商类名，可抵抗 PyNvVideoCodec 版本间的命名漂移。
2. 元数据读取边界归一化：decode_frames_pynvvideocodec 中包裹 _read_source_metadata 调用，命中 PyNvVCException* 则抛出 ValueError("Invalid or unsupported video file.") 并保留 cause，其余异常原样重抛。该路径覆盖截断 MKV、伪格式等在打开 / 扫流阶段的失败。
3. 帧解码边界归一化：_decode_to_pinned_host 中把 get_decoder 与 get_batch_frames_by_index 包进同一 try/except，归一化范围扩大为 PyNvVCException* + IndexError。IndexError 是 H200 实测确认的部分损坏流在批量取帧时厂商抛出的信号；把 get_decoder 纳入 try 还使解码器槽重建失败也被归一化。
4. 不改动 API 层：vLLM 已有 ValueError → BadRequestError → HTTP 400 的映射，本 PR 只在媒体边界做翻译，确保 GPU 解码器不可用等服务器侧异常继续以原始状态暴露，避免被伪装成客户端错误。

5. 测试配套：tests/multimodal/test_video.py 新增 CUDA 门控的 test_pynvvideocodec_corrupted_videos_raise_value_error，用仓库 corrupted.mp4 资产覆盖截断打开失败、部分损坏解码失败、随后有效解码恢复三条路径；tests/multimodal/media/test_video.py 新增 test_pynvvideocodec_unrelated_error_propagates，用 fake nvc 模块验证无关 RuntimeError 不被归一化。

关键文件：

- vllm/multimodal/video.py（模块 视频解码；类别 source；类型 core-logic；符号 _pynvvideocodec_exception_types, _decode_to_pinned_host, decode_frames_pynvvideocodec）：全部生产代码变更所在：新增 PyNvVCException 类型探测辅助函数，并在元数据读取与帧解码两个媒体边界做异常归一化，是 500→400 修复的核心。
- tests/multimodal/test_video.py（模块 回归测试；类别 test；类型 test-coverage；符号 test_pynvvideocodec_corrupted_videos_raise_value_error）：CUDA 门控的真实 PyNvVideoCodec 集成测试，用仓库 corrupted.mp4 覆盖截断打开失败、部分损坏解码失败与有效解码恢复三条路径，是本次修复的主回归防线。
- tests/multimodal/media/test_video.py（模块 回归测试；类别 test；类型 test-coverage；符号 test_pynvvideocodec_unrelated_error_propagates, FakePyNvVCException, raise_unrelated_error）：CPU 侧回归测试，用 fake nvc 模块 + monkeypatch 验证无关 RuntimeError 不被归一化，锁定了异常翻译的边界，防止未来把服务器错误误报为客户端错误。

关键符号：_pynvvideocodec_exception_types, decode_frames_pynvvideocodec, _decode_to_pinned_host, test_pynvvideocodec_corrupted_videos_raise_value_error, test_pynvvideocodec_unrelated_error_propagates

关键源码片段

vllm/multimodal/video.py

全部生产代码变更所在：新增 PyNvVCException 类型探测辅助函数，并在元数据读取与帧解码两个媒体边界做异常归一化，是 500→400 修复的核心。

```
def _pynvvideocodec_exception_types(nvc) -> tuple[type[Exception], ...]:
    # 动态收集 PyNvVideoCodec 导出的所有 PyNvVCException* 异常类。
    # 不硬编码类名，避免厂商在不同版本改名或新增子类时漏掉。
    return tuple(
        exception_type
        for name in dir(nvc)
        if name.startswith("PyNvVCException")
        and isinstance((exception_type := getattr(nvc, name)), type)
        and issubclass(exception_type, Exception)
    )


@classmethod
def _decode_to_pinned_host(
    cls,
```

```
    file_path: str,
    frame_idx: list[int],
    nvc,
) -> npt.NDArray:
    # 帧解码边界：get_decoder 负责复用 / 重建解码器槽，
    # get_batch_frames_by_index 负责批量取帧，两者都可能抛厂商异常。
    with cls._borrow_decoder_slot() as decoder_slot:
        stream = decoder_slot.stream
        with cls._torch_stream_context(stream):
            try:
                decoder = decoder_slot.get_decoder(
                    file_path, nvc, device_index=cls._DEVICE_INDEX
                )
                decoded_frames = decoder.get_batch_frames_by_index(frame_idx)
            except Exception as exc:
                # 只归一化厂商异常与部分损坏流特有的 IndexError，
                # 其余异常（如 GPU 解码器不可用）原样传播，避免掩盖服务器故障。
                if not isinstance(
                    exc,
                    _pynvvideocodec_exception_types(nvc) + (IndexError,),
                ):
                    raise
                raise ValueError("Invalid or unsupported video file.") from exc
            # 后续帧形状校验、显存到主机拷贝逻辑保持不变……

@classmethod
def decode_frames_pynvvideocodec(
    cls,
    data: bytes,
    target: VideoTargetMetadata,
    **kwargs,
) -> tuple[npt.NDArray, VideoSourceMetadata, list[int], list[int]]:
    temp_fd, temp_path = tempfile.mkstemp(suffix=".mp4")
    try:
        with os.fdopen(temp_fd, "wb") as temp_file:
            temp_file.write(data)
        # 元数据读取边界：截断 / 伪格式视频在此阶段抛 PyNvVCException。
        try:
            gpu_source = cls._read_source_metadata(temp_path, nvc)
        except Exception as exc:
            if not isinstance(exc, _pynvvideocodec_exception_types(nvc)):
                raise
            raise ValueError("Invalid or unsupported video file.") from exc
        _check_frame_pixel_limit(gpu_source.width, gpu_source.height)
        # 帧采样与解码池分配逻辑保持不变……
    finally:
        with suppress(FileNotFoundError):
            os.unlink(temp_path)
```

tests/multimodal/test_video.py

CUDA 门控的真实 PyNvVideoCodec 集成测试，用仓库 corrupted.mp4 覆盖截断打开失败、部分损坏解码失败与有效解码恢复三条路径，是本次修复的主回归防线。

```
@pytest.mark.skipif(not current_platform.is_cuda(), reason="Requires CUDA")
def test_pynvvideocodec_corrupted_videos_raise_value_error():
    # 用仓库真实损坏资产覆盖两条失败路径，并用有效视频验证解码器可恢复。
    valid_video = create_long_gop_video(num_frames=2, width=64, height=64)
    corrupted_video = (ASSETS_DIR / "corrupted.mp4").read_bytes()
    malformed_video = corrupted_video[:128]

    # 记录并重置解码器槽位状态，确保测试隔离。
    old_slots = PyNvVideoCodecVideoBackend._decoder_slots
    old_active_slots = PyNvVideoCodecVideoBackend._active_decoder_slots
    old_cond = PyNvVideoCodecVideoBackend._decoder_slot_cond
    old_max_slots = PyNvVideoCodecVideoBackend._max_decoder_slots
    try:
        PyNvVideoCodecVideoBackend._decoder_slots = []
        PyNvVideoCodecVideoBackend._active_decoder_slots = 0
        PyNvVideoCodecVideoBackend._decoder_slot_cond = threading.Condition()
        PyNvVideoCodecVideoBackend._max_decoder_slots = None

        loader = VIDEO_LOADER_REGISTRY.load(PYNVVIDEOCODEC_VIDEO_BACKEND)

        # 截断文件：打开 / 元数据阶段失败，cause 为 PyNvVCException。
        with pytest.raises(
            ValueError, match=r"^Invalid or unsupported video file\.$"
        ) as malformed_exc:
            loader.load_bytes(malformed_video, num_frames=1, hw_decoders=1)
        assert malformed_exc.value.__cause__ is not None

        # 完整损坏文件：批量解码阶段失败，cause 为厂商 IndexError。
        with pytest.raises(
            ValueError, match=r"^Invalid or unsupported video file\.$"
        ) as exc_info:
            loader.load_bytes(corrupted_video, num_frames=-1, hw_decoders=1)
        assert exc_info.value.__cause__ is not None

        # 错误后同一后端仍能成功解码有效视频，验证解码器槽未被污染。
        frames, _ = loader.load_bytes(valid_video, num_frames=1, hw_decoders=1)
        assert frames.shape[0] == 1
    finally:
        # 恢复解码器槽位状态，避免影响同进程其他测试。
        PyNvVideoCodecVideoBackend._decoder_slots = old_slots
        PyNvVideoCodecVideoBackend._active_decoder_slots = old_active_slots
        PyNvVideoCodecVideoBackend._decoder_slot_cond = old_cond
        PyNvVideoCodecVideoBackend._max_decoder_slots = old_max_slots
```

评论区精华

review 的核心交锋集中在测试策略上。Isotr0py 首轮要求用仓库真实损坏资产替代合成字节（"Can we use the corrupted video (tests/multimodal/assets/corrupted.mp4) for test instead?"），随后澄清要的是端到端真实触发："I mean some e2e tests with corrupted video to trigger exception instead of self-contained ut with monkeypatch."。作者分三步回应：先改用 corrupted.mp4 资产，再新增 CUDA 门控的真实 PyNvVideoCodec 集成测试（覆盖损坏流映射 ValueError 与有效解码恢复），最终删除冗余的 mock invalid-video 单测，仅保留验证无关异常传播的 CPU 测试。实现决策上，归一化范围被精确限定为已导出的 PyNvVCException* 加部分损坏流特有的 IndexError，保留 cause，无关异常如 GPU 解码器不可用的 RuntimeError 必须原样传播。

- 用真实 corrupted.mp4 资产替代合成字节 (testing): 作者按建议改用真实资产，并进一步演进为 CUDA 门控的真实后端集成测试。
- mock 单测 vs 端到端真实触发 (testing): 作者新增 CUDA 门控集成测试覆盖损坏视频两条失败路径，并最终移除冗余的 mock invalid-video 单测，仅保留验证无关异常传播的 CPU 测试。
- 归一化边界：PyNvVCException* 与 IndexError 的取舍 (design): 边界被最终代码与测试锁定：CPU 测试验证 RuntimeError 不被归一化，CUDA 测试验证两侧 ValueError 均带 cause。

风险与影响

- 风险：
  1. 厂商异常命名依赖：_pynvvideocodec_exception_types 运行时动态探测 dir(nvc)，若 PyNvVideoCodec 新版改变异常前缀或改为非类导出，探测落空会让坏输入退回 500。已有测试锁定当前版本行为，但上游升级时需要留意。
  2. IndexError 匹配偏宽：try 块覆盖 get_decoder + get_batch_frames_by_index 后，任何源自这两处调用的 IndexError 都会被翻译为 Invalid video；若未来重排代码引入新的索引逻辑，可能误归一化。当前范围内可控。
  3. 解码器槽重建失败语义风险：get_decoder 纳入 try 后，解码器槽重建 / 显存分配失败也会被归一化为 400，这类失败本质可能是服务器资源问题而非输入问题。
  4. API 层回归未自动化：HTTP 400 行为仅在 B200 上手工验证，没有 API 层自动化测试，后续重构可能回归。
- 影响：
  1. API 用户：坏视频请求响应从 500 变为 400，错误语义清晰，客户端可区分输入错误与服务故障。
  2. 监控运营：错误日志不再因用户输入刷 InternalServerError，告警噪音下降。
  3. 系统改动面：仅影响 pynvvideocodec 后端路径，DeepStream、PyAV、OpenCV 后端不受影响；解码器复用语义未变，错误后仍可继续服务有效请求（集成测试覆盖）。
  4. 团队：为 GPU 厂商解码器的输入校验提供统一边界范例，后续可推广到其他硬件解码路径。 - 风险标记：厂商异常类型动态探测，IndexError 匹配偏宽，解码器槽重建误归一

化，API 层回归未自动化

关联脉络

- PR #51139 [Bugfix][Multimodal] Invalidate retained PyNvVideoCodec decoder after failure: 同属 PyNvVideoCodec 后端错误处理系列：51139 处理失败后解码器实例状态污染，本 PR 处理输入异常语义与解码器复用路径的归一化，二者共同完善 GPU 视频解码的故障恢复。
- PR #46747 [Bugfix][V1][Multimodal] Recover from P0/P1 processor cache drift: 多模态路径从崩溃 /500 转向可恢复、语义正确的错误处理，与本 PR 的 500→400 归一化属于同一演进方向。