

PR #51076 完整报告

vllm-project/vllm

[Bugfix][Multimodal] Fix PyNvVideoCodec video backend returning NCHW instead of NHWC

合并时间: 2026-08-08 12:49

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51076>

执行摘要

- 一句话: 修复 PyNvVideoCodec 视频后端 NCHW/NHWC 布局错乱
- 推荐动作: 值得精读的小型 bugfix, 是“实现与契约背离导致整条链路失败”的典型案列: docstring 声明 NHWC、其余后端均遵守、唯独一个后端悄悄违反。值得关注的决策: 1) 不盲目删除 permute, 而是用通道轴检测兼容 PyNvVideoCodec 版本间布局漂移 (源自 Isotr0py 对 #44465 内部讨论的引用); 2) 用纯 CPU 构造张量的参数化单测直接钉住回归点, 避免单测依赖 GPU 专有后端; 3) 修复后对 mp4/webm/mov 三种容器格式做了端到端验证, 补齐了“单测覆盖逻辑、e2e 覆盖行为”的验证闭环。

功能与动机

PR body 明确指出 video-loader 契约要求返回 NHWC,

`PyNvVideoCodecVideoBackendMixin._decode_to_pinned_host` 的 docstring 也声明帧是 "NHWC RGB format ... identical to the OpenCV/PyAV backends", 且 OpenCV/PyAV/TorchCodec/DeepStream 各后端均返回 NHWC; 唯独 `pynvvc` 路径多做了 `permute(0, 3, 1, 2)`, 把 `torch.stack` 得到的 `(N, H, W, C)` 翻转成 `(N, C, H, W)`, 任何通过 HF video processor 逐帧物化 PIL 图像模型都会失败: `BadRequestError: Cannot handle this data type: (1, 1, 320), lu1` (在 gemma-4-31b 上复现)。mgoin 在 review 中追问该 bug 是否一直存在, 作者确认 "it's been broken since the PyNvVideoCodec integration in #44465"。

实现拆解

1. 根因定位: 在 `vllm/multimodal/video.py` 的 `PyNvVideoCodecVideoBackendMixin._decode_to_pinned_host` 中, `torch.stack(torch_frames)` 已把逐帧 HWC 图像堆叠为 `(N, H, W, C)`; 原有代码多余的 `permute(0, 3, 1, 2)` 将布局翻转为 NCHW, 与方法 docstring 以及其余各视频后端行为相悖。
2. 方案演进: 第一版提交直接删除 `permute` 并保留 `contiguous()`; 经 Isotr0py 审查提醒 (PyNvVideoCodec 存在 NHWC 与 NCHW 两种布局能力, 行为随版本变化), 改为新增模块级函数 `_pynvvc_frames_to_nhwc`: 当 `shape[-1] != 3 and shape[-3] == 3` 时判定为 NCHW 并 `permute(0, 2, 3, 1)` 转回 NHWC, 末尾统一 `contiguous()` 保证后续 pinned 内存异步拷贝 (`copy_non_blocking`) 安全; 该函数固化在 `vllm/multimodal/video.py` 模块顶层, 调用点替换原 `permute`。

3. 测试配套：在 tests/multimodal/media/test_video.py 新增 test_pynvvc_frames_normalized_to_nhwc，用 @pytest.mark.parametrize("layout", ["nhwc", "nchw"]) 覆盖两种输入布局，断言输出形状 (N, H, W, C)、张量连续性与内容保真；通过 pytest.importorskip("torch") 和 cpu_test 标记保持纯 CPU 可运行，不依赖 GPU 与 PyNvVideoCodec 专有依赖。
4. 端到端验证：作者在 B200 + gemma-4-31b (NVFP4) + VLLM_VIDEO_LOADER_BACKEND=pynvvideocodec 环境下对 mp4 / webm / mov 三种容器格式的原生 video_url 请求验证：修复前全部 HTTP 400，修复后三种格式均输出正确的 SMPTE 彩条描述，且 load_bytes 返回形状确认为 (N, H, W, 3)。本 PR 无配置、schema 或部署配套改动。

关键文件：

- vllm/multimodal/video.py (模块 视频加载；类别 source；类型 core-logic；符号 _pynvvc_frames_to_nhwc, _decode_to_pinned_host)：核心修复文件。
PyNvVideoCodecVideoBackendMixin._decode_to_pinned_host 中错误的 permute(0, 3, 1, 2) 把 NHWC 帧翻转为 NCHW，违反 video-loader 契约并导致 gemma-4 等模型原生视频请求 HTTP 400。新增 _pynvvc_frames_to_nhwc 通过通道轴检测兼容 NHWC/NCHW 两种输入，统一归一化到连续 NHWC。
- tests/multimodal/media/test_video.py (模块 视频测试；类别 test；类型 test-coverage；符号 test_pynvvc_frames_normalized_to_nhwc)：新增参数化 CPU 单测 test_pynvvc_frames_normalized_to_nhwc，覆盖 NHWC/NCHW 两种输入布局，钉住回归点（形状、连续性、内容保真），且不依赖 GPU 与 PyNvVideoCodec 专有依赖，可在常规 CI 运行。

关键符号：_pynvvc_frames_to_nhwc, _decode_to_pinned_host,
test_pynvvc_frames_normalized_to_nhwc

关键源码片段

vllm/multimodal/video.py

核心修复文件。PyNvVideoCodecVideoBackendMixin._decode_to_pinned_host 中错误的 permute(0, 3, 1, 2) 把 NHWC 帧翻转为 NCHW，违反 video-loader 契约并导致 gemma-4 等模型原生视频请求 HTTP 400。新增 _pynvvc_frames_to_nhwc 通过通道轴检测兼容 NHWC/NCHW 两种输入，统一归一化到连续 NHWC。

```
def _pynvvc_frames_to_nhwc(frames):
    """将 PyNvVideoCodec 堆叠帧批量归一化为连续 NHWC 布局。

    PyNvVideoCodec 的逐帧布局在不同版本间有变化 (HWC vs CHW) ,
    因此这里检测通道轴而非假设固定顺序。NHWC 是其他视频后端
    返回的布局，也是 HF 视频处理器期望的布局。

    Args:
        frames: ``(N, ?, ?, ?)`` 的 uint8 张量，NHWC 或 NCHW 顺序均可。
```

Returns:

```

    归一化后连续排列的 ``(N, H, W, C)`` 张量。
    """
    # NHWC 布局下最后一维恒为 3 通道；若末维非 3 而倒数第三维为 3，
    # 则判定输入为 NCHW，转置回 NHWC（内容保持不变）
    if frames.shape[-1] != 3 and frames.shape[-3] == 3:
        frames = frames.permute(0, 2, 3, 1) # NCHW -> NHWC
    # contiguous() 保证后续 pinned 内存异步拷贝（copy_non_blocking）安全；
    # 输入已连续时是 no-op，无额外开销
    return frames.contiguous()

    # ...
    device_frames = torch.stack(torch_frames) # 逐帧 HWC 堆叠为 (N, H, W, C)
    if device_frames.ndim != 4:
        raise ValueError(
            "PyNvVideoCodec returned frames with unexpected shape "
            f"{tuple(device_frames.shape)}"
        )
    # 统一到 NHWC：旧实现在这里错误地 permute 成 NCHW，破坏 video-loader 契约
    device_frames = _pynvvc_frames_to_nhwc(device_frames)
    host_frames = torch.empty(
        device_frames.shape,
        dtype=device_frames.dtype,
        device="cpu",
        pin_memory=True,
    )
    # 异步拷贝到 pinned host 内存，再同步流，最后转 numpy 返回
    host_frames.copy_(device_frames, non_blocking=True)
    stream.synchronize()
    host_array = host_frames.numpy()

```

tests/multimodal/media/test_video.py

新增参数化 CPU 单测 `test_pynvvc_frames_normalized_to_nhwc`，覆盖 NHWC/NCHW 两种输入布局，钉住回归点（形状、连续性、内容保真），且不依赖 GPU 与 PyNvVideoCodec 专有依赖，可在常规 CI 运行。

```

@pytest.mark.parametrize("layout", ["nhwc", "nchw"])
def test_pynvvc_frames_normalized_to_nhwc(layout):
    """无论解码器输出哪种逐帧布局，PyNvVideoCodec 帧批量都应归一化到
    NHWC，保证 HF 视频处理器（逐帧构造 PIL 图像）拿到与其他后端一致的形状。"""
    torch = pytest.importorskip("torch") # 保持纯 CPU 可运行，不依赖 GPU 与专有后端

    n, h, w, c = 4, 5, 6, 3
    nhwc = torch.arange(n * h * w * c, dtype=torch.uint8).reshape(n, h, w, c)
    # NCHW 输入由 NHWC 基准转置而来，保证内容可精确对照
    frames = nhwc if layout == "nhwc" else nhwc.permute(0, 3, 1, 2).contiguous()

    out = _pynvvc_frames_to_nhwc(frames)

    assert out.shape == (n, h, w, c) # 形状统一为 NHWC

```

```
assert out.is_contiguous() # 保证 pinned 异步拷贝路径可用
assert torch.equal(out, nhwc) # 内容保真：未损坏或错位
```

评论区精华

Review 中两条关键反馈推动了最终实现形态：

- mgoin: "I believe you, but has this just been broken forever? Could you add a unit or e2e test that would catch this?" —— 促使作者补充了纯 CPU 的参数化单测，并确认 bug 从 #44465 集成起就存在。
- Isotr0py: 指出 "PyNvVideoCodec will always convert NHWC to NCHW" (引用 #44465 内部讨论 r3391283504)，建议不要只删 permute 而是回退到同时兼容 NHWC 与 NCHW 的原始实现。作者采纳，在 Issue 评论中说明: "Rather than swapping one hardcoded layout for another, I now detect the channel axis and normalize to NHWC... NHWC always has `shape[-1] == 3`", 最终落地为 `_pynvvc_frames_to_nhwc`。两人最终均 approve，无未解决疑虑。
- PyNvVideoCodec 帧布局是否需要兼容 NHWC 与 NCHW 两种顺序 (design): 作者采纳，实现 `_pynvvc_frames_to_nhwc`: 以 `shape[-1] == 3` 判定 NHWC、`shape[-3] == 3` 判定 NCHW，统一归一化到 NHWC，兼容两种布局而非硬编码单一顺序。
- bug 是否长期存在以及是否需要补充测试 (testing): 作者确认 bug 自 #44465 PyNvVideoCodec 集成起就存在，并新增纯 CPU 参数化单测 `test_pynvvc_frames_normalized_to_nhwc` 覆盖 NHWC/NCHW 两种输入；端到端行为由作者在 B200 上手动验证。

风险与影响

- 风险：
 - 布局推断依赖 3 通道假设: `_pynvvc_frames_to_nhwc` 以 `shape[-1] == 3` 判定 NHWC；若未来出现非 RGB 帧（灰度 C=1 或 4 通道），检测可能误判或静默跳过转置。当前 video-loader 契约固定 RGB，风险有限，但该启发式最好固化到函数契约中。
 - 回归面：改动仅涉及 `vllm/multimodal/video.py` 中 `pynvvc` 分支的单一调用点，其他后端不受影响；但也意味着该缺陷在常规 CI GPU 测试中不易暴露。
 - 测试缺口：单测只覆盖纯张量归一化逻辑，未覆盖真实 `decode_frames_pynvvideocodec` 解码路径（依赖 GPU 与 PyNvVideoCodec 专有依赖），CI 无法自动回归该端到端场景，后续布局再次漂移仍可能漏出。
 - 性能：`contiguous()` 对已连续张量是 no-op，`permute` 仅在检测到 NCHW 时执行，无实质开销。
- 影响：
 - 用户影响：恢复 `VLLM_VIDEO_LOADER_BACKEND=pynvvideocodec` 下原生 video 请求的可用性，对 gemma-4 等逐帧 PIL 处理模型是直接修复；使用其他视频后端或非逐帧 processor 的用户无行为变化。
 - 系统影响：仅触及多模态视频加载管线中 `pynvvc` 分支，无接口、配置或部署变更。

- 团队影响：改动小、聚焦 (+39/-1)，风险低；新函数 docstring 把 NHWC 契约显式文档化，为后续后端实现提供了参考。
- 风险标记：布局推断依赖 3 通道假设，真实解码路径缺 CI 覆盖，回归面集中于单一后端

关联脉络

- PR #44465 [上下文未提供原标题] PyNvVideoCodec video backend integration: PyNvVideoCodec 后端集成 PR，本 PR 评论区确认 bug 源头为其中引入的 `permute(0, 3, 1, 2)`；Isotr0py 引用了该 PR 内部讨论 r3391283504 说明布局行为随版本变化。
- PR #31925 [上下文未提供原标题] Original zero-copy PyNvVideoCodec IPC integration (DRAFT): PR body 说明这是 PyNvVideoCodec 的最初零拷贝 IPC 集成原型 (DRAFT / DO-NOT-REVIEW)，未触及本帧布局 bug，属于同一功能线的上游起点。
- PR #47344 [上下文未提供原标题] IPC-calc refactor: PR body 提及该 PR 重构了 IPC 计算逻辑，但不涉及帧布局问题，与本 PR 同属 PyNvVideoCodec 集成演进线。
- PR #51408 [1/N] Harden Transformers modelling backend multi-modal path: 同属多模态处理链加固方向，解决 HF processor/backends 契约不一致导致的崩溃问题，与本 PR 修复视频帧布局契约违规是同一质量收敛趋势。