PR #41532 完整报告

vllm-project/vllm

[ROCm][CI] Gate incompatible HF references on Transformers v5

合并时间：2026-06-16 20:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41532

执行摘要

- 一句话：适配 Transformers v5 的多模态处理和 CI 测试兼容性
- 推荐动作：本 PR 值得精读，尤其是 _restore_fp32_rope_buffers 的 _apply 覆盖模式和多模态处理器向 HF 上游对齐的设计思路。对于关注多模态模型兼容性和 ROCm CI 稳定的开发者，建议仔细 review musicflamingo.py 和 audioflamingo3.py 的变更。同时，注意测试注册表中的版本门控策略，未来维护时需保持更新。

功能与动机

由于 HuggingFace Transformers v5 移除了 is_torch_fx_available 等函数，导致依赖 remote code 的 MiniCPM4 等模型无法加载；HyperCLOVAX 也因 v5 不再暴露默认 RoPE 配置而失败。此外，AudioFlamingo3 和 MusicFlamingo 的 HF processor 在 v5 中已有原生实现，vLLM 应转为调用上游 processor 而非维护本地副本。详见 PR body 和关联 Issue #44561。

实现拆解

1. 测试注册表版本门控（tests/models/registry.py）：为 MiniCPM4ForCausalLM 和 HyperCLOVAXForCausalLM 添加 min_transformers_version 和 max_transformers_version 限制，并附上 transformers_version_reason 说明不兼容原因，确保这些模型仅在兼容的 Transformers 版本上运行测试。
2. AudioFlamingo3 处理器重构（vllm/model_executor/models/audioflamingo3.py）：将 _call_hf_processor 改为调用 super()._call_hf_processor() 以直接使用上游 HF processor，移除本地的 prompt 编辑逻辑和手动的 tokenization；同时将 get_supported_mm_limits 从 {"audio": None} 改为 {"audio": 1} 以限制单音频输入；在 _encode_audio_features 中修正 input_features 的数据类型转换位置。
3. MusicFlamingo 模型增强（vllm/model_executor/models/musicflamingo.py）：新增 _restore_fp32_rope_buffers 方法和 _apply 覆盖，确保在 nn.Module._apply（如 to() 或 half()）后 RoPE 缓存（inv_freq, position_angles）始终以 fp32 精度重计算；修正 forward 中的时间窗口计算，从基于 batch_positions 改为基于 window_starts 和 window_duration；将 MusicFlamingoFeatureInputs.rote_timestamps 类型由 torch.Tensor 改为 torch.Tensor | None；为 MusicFlamingoProcessingInfo.get_data_parser 指定 audio_resample_method="soxr"。

4. 音频重采样支持 soxr（vllm/multimodal/audio.py）：新增 resample_audio_soxr 函数，利用 soxr 库进行高质量重采样；扩展 AudioResampler 类的 method 参数支持 "soxr" 选项。
5. 测试适配（多个测试文件）：更新 test_audioflamingo3.py 和 test_musicflamingo.py 的 mock，移除旧的 monkeypatch 方式，改为通过 ctx.call_hf_processor 的 side_effect 模拟 HF processor 调用；增加对 feature_attention_mask 的断言；在 test_musicflamingo.py 生成测试中加入预热推理以提高 ROCm 下首次编译的稳定性。

关键文件：

- vllm/model_executor/models/musicflamingo.py（模块 模型执行器；类别 source；类型 data-contract；符号 _restore_fp32_rope_buffers, _apply, _call_hf_processor, _build_audio_timestamps）：核心模型文件，包含 RoPE 缓存恢复(_restore_fp32_rope_buffers, _apply)、时间窗口计算修正、rote_timestamps 类型变更，以及默认音频重采样方法改为 soxr。
- vllm/multimodal/audio.py（模块 多模态；类别 source；类型 dependency-wiring；符号 resample_audio_soxr）：新增 soxr 音频重采样函数和 AudioResampler 扩展，为多模态模型提供更多重采样选择。
- vllm/model_executor/models/audioflamingo3.py（模块 模型执行器；类别 source；类型 data-contract）：重构 _call_hf_processor 委托给父类，移除本地复制逻辑；修正音频限制为单音频。
- tests/models/multimodal/processing/test_musicflamingo.py（模块 MusicFlamingo；类别 test；类型 test-coverage；符号 call, test_musicflamingo_chunk_counting_uses_rote_timestamps, test_musicflamingo_chunk_counting_without_rote_timestamps, mock_base_call）：测试文件，调整 mock 以适配 HF processor，移除旧 monkeypatch，增加 feature_attention_mask 断言。
- tests/models/multimodal/processing/test_audioflamingo3.py（模块 AudioFlamingo3；类别 test；类型 test-coverage；符号 call, _tokenize, mock_base_call）：测试文件，更新 mock 和断言，使用 ctx.call_hf_processor 模拟调用。
- tests/models/multimodal/generation/test_musicflamingo.py（模块 MusicFlamingo；类别 test；类型 test-coverage；符号 load_expected_fixture, test_single_generation）：生成测试文件，添加预热推理和 fixture 加载函数，提升 ROCm 稳定性。
- tests/models/registry.py（模块 注册表；类别 test；类型 test-coverage）：测试注册表，为不兼容模型添加版本门控和原因。
- vllm/multimodal/parse.py（模块 多模态；类别 source；类型 core-logic）：微调配置键，可能涉及多模态数据解析适配。
- tests/models/language/generation/test_common.py（模块 通用；类别 test；类型 test-coverage）：语言模型生成测试通用调整，可能与版本门控配合。
- setup.py（模块 构建脚本；类别 config；类型 configuration）：构建配置，可能添加了 soxr 依赖。

关键符号：_restore_fp32_rope_buffers, _apply, _call_hf_processor, resample_audio_soxr, _build_audio_timestamps

关键源码片段

vllm/model_executor/models/musicflamingo.py

核心模型文件，包含 RoPE 缓存恢复 (_restore_fp32_rope_buffers, _apply)、时间窗口计算修正、rote_timestamps 类型变更，以及默认音频重采样方法改为 soxr。

```
# vllm/model_executor/models/musicflamingo.py
class MusicFlamingoRotaryEmbedding(nn.Module):
    # ... 初始化省略 ...

    def _restore_fp32_rope_buffers(self) -> None:
        # 重算 inv_freq 和 position_angles，确保在 _apply (half/to) 后保持 fp32
        rope_init_fn: Callable = self.compute_default_rope_parameters
        if self.rope_type != "default":
            rope_init_fn = ROPE_INIT_FUNCTIONS[self.rope_type]
        inv_freq, self.attention_scaling = rope_init_fn(
            self.config, self.inv_freq.device
        )
        self.inv_freq = inv_freq
        self.original_inv_freq = inv_freq.clone()
        self.position_angles = self._compute_position_angles(inv_freq)

    def _apply(self, fn):
        # 拦截 nn.Module._apply（被 to()、half() 等调用），
        # 在父类转换后立即恢复 fp32 缓存
        super()._apply(fn)
        self._restore_fp32_rope_buffers()
        return self

    @torch.no_grad()
    def forward(self, timestamps: Tensor, seq_len: int) -> tuple[Tensor, Tensor]:
        # 改用 window_starts 计算频率，使时间窗口对齐音频帧步长
        window_starts = timestamps[:, 0].to(
            device=self.inv_freq.device,
            dtype=self.inv_freq.dtype,
        )
        window_duration = self.config.audio_frame_step * 4 * seq_len
        window_positions = (
            torch.round(window_starts / window_duration) / self.max_seq_len_cached
        )
        window_freqs = window_positions.unsqueeze(-1) * self.inv_freq
        window_freqs = torch.repeat_interleave(window_freqs, 2, dim=-1)
        # ... 后续计算省略 ...
```

vllm/multimodal/audio.py

新增 soxr 音频重采样函数和 AudioResampler 扩展，为多模态模型提供更多重采样选择。

```
# vllm/multimodal/audio.py
def resample_audio_soxr(
```

```
    audio: npt.NDArray[np.floating],
    *,
    orig_sr: float,
    target_sr: float,
) -> npt.NDArray[np.floating]:
    """使用 soxr 库对音频数据进行重采样。"""
    orig_sr_int = int(round(orig_sr))
    target_sr_int = int(round(target_sr))
    if orig_sr_int == target_sr_int:
        return audio
    # 多声道处理：逐声道递归
    if audio.ndim == 2:
        return np.stack(
            [resample_audio_soxr(ch, orig_sr=orig_sr, target_sr=target_sr) for ch in audio],
            axis=0,
        )
    return soxr.resample(audio, orig_sr_int, target_sr_int)


class AudioResampler:
    def __init__(
        self,
        target_sr: float | None = None,
        method: Literal["pyav", "scipy", "soxr"] = "pyav", # 新增 "soxr" 选项
    ):
        ...
    def resample(self, audio, *, orig_sr):
        ...
        elif self.method == "soxr":
            return resample_audio_soxr(audio, orig_sr=orig_sr, target_sr=self.target_sr)
```

vllm/model_executor/models/audioflamingo3.py

重构 _call_hf_processor 委托给父类，移除本地复制逻辑；修正音频限制为单音频。

```
# vllm/model_executor/models/audioflamingo3.py
class
AudioFlamingo3MultiModalProcessor(BaseMultiModalProcessor[AudioFlamingo3ProcessingInfo]):
    def _call_hf_processor(
        self,
        prompt: str,
        mm_data: dict[str, object],
        mm_kwargs: Mapping[str, Any],
        tok_kwargs: Mapping[str, object],
    ) -> BatchFeature:
        # 复制 mm_data 避免修改原始输入
        processor_mm_data = dict(mm_data)
        audios = processor_mm_data.pop("audios", None)
        if audios is not None:
            processor_mm_data["audio"] = audios

        # 直接调用父类委托给 HF processor，移除手动 tokenization
```

```
        outputs = super()._call_hf_processor(
            prompt=prompt,
            mm_data=processor_mm_data,
            mm_kwargs=mm_kwargs,
            tok_kwargs=tok_kwargs,
        )
        # 统一字段名
        if "input_features_mask" in outputs:
            outputs["feature_attention_mask"] = outputs.pop("input_features_mask")
        # 仅在有音频数据时计算 chunk_counts
        audio_data = processor_mm_data.get("audio")
        if audio_data is None:
            return outputs
        # ... 计算 chunk_counts 逻辑 ...
        outputs["chunk_counts"] = torch.tensor(chunk_counts, dtype=torch.long)
        return outputs
```

评论区精华

- gemini-code-assist[bot] 指出逻辑 bug：添加 transformers_version_reason 会导致 check_transformers_version 始终跳过测试，因为 is_reason_valid 在有 reason 时为 False。作者已修复此问题。
  - hmellor 强调 MusicFlamingo 在 Transformers 5.5 已原生支持，不应跳过测试。PR 实际并未跳过，而是通过适配确保在 v5 下正常运行。
- eustlb 确认 processor 重构方向正确：'Looks like a lot of the confusion here comes from the fact that the vLLM PR was merged before the transformers one on which it depended'，并指出应使用 super()._call_hf_processor，获得作者采纳。
- 作者回应版本范围质疑：MiniCPM4 的版本限制 4.56-4.57 是故意的，因为当前 Transformers 已是 v5，而 remote code 在 v5 中因移除 is_torch_fx_available 而失败，上游兼容请求已被关闭。
  - transformers_version_reason 导致测试始终跳过 (correctness): 作者在后续提交中修复了此逻辑。
  - MiniCPM4 版本范围是否过于严格 (question): 作者解释该范围是故意的，因为当前 Transformers 最新版已是 v5，且 MiniCPM4.1 的 remote code 在 v5 中因移除 is_torch_fx_available 而损坏，上游兼容请求已关闭。
  - MusicFlamingo 在 v5.5 已原生支持 (correctness): PR 并未跳过 MusicFlamingo 测试，而是适配了 processor 以使用原生 HF 实现。
  - Processor 应使用上游 HF 调用 (design): 作者采纳建议，重构了 processor 实现。

风险与影响

- 风险：
  - 测试跳过覆盖不足：HyperCLOVAX 和 MiniCPM4 的版本门控可能导致在特定 Transformers 版本（如 v4.56-4.57）之外测试被跳过，若未来 v4.x 有更新，兼容性可

能被忽略。

- 可选依赖降级路径：新增 soxr 为可选依赖，若未安装则回退到 scipy 或 pyav，但 MusicFlamingoProcessingInfo 默认指定 audio_resample_method="soxr"，需要确保下游方安装或优雅处理缺失。
- RoPE 类型处理：_restore_fp32_rope_buffers 中对非 default 的 rope_type 依赖 ROPE_INIT_FUNCTIONS[self.rope_type]，若 v5 中该字典不含对应类型，可能导致 KeyError。
- Processor 重构回归：AudioFlamingo3._call_hf_processor 从手动处理改为委托给父类，如果上游 HF processor 行为变化，vLLM 的行为可能不一致。

- 影响：
  - 用户：使用 Transformers v5 的用户现在可以正常加载 MiniCPM4、HyperCLOVAX、AudioFlamingo3 和 MusicFlamingo 模型，之前因 remote code 错误而失败的多模态任务恢复运行。
  - 系统：音频重采样新增 soxr 方法，提供高质量选项，但依赖可选。多模态处理器代码量精简，减少本地维护成本。
  - CI：ROCm 环境下的多模态生成测试更加稳定，通过预热推理避免了首次编译的不确定性。测试版本门控确保不兼容模型不被测试，减少失败噪音。
  - 团队：需要审计所有多模态模型，确保 transformers_version_reason 配置正确；未来升级 Transformers 时需及时更新版本范围。
  - 风险标记：测试跳过覆盖不足，可选依赖降级路径，RoPE 类型处理，核心路径变更

关联脉络

- PR #39011 WIP: AF-Next 集成：review 中 lashahub 提到此 PR，表示等当前 PR 合并后 rebase #39011，避免重复的 Flamingo processor/RoTE 修复。
- PR #44561 Removal of is_torch_fx_available in v5.0 breaks trust_remote_code models: 关联 Issue，描述了 Transformers v5 移除函数的 breaking change，本 PR 的版本门控正是为此。