PR #48729 完整报告

vllm-project/vllm

[Bugfix][GLM4V] Fix video dummy profiling and memory usage

合并时间：2026-07-18 01:44

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48729

执行摘要

- 一句话：修复 GLM4V 虚拟视频帧搜索与内存占用过高
- 推荐动作：建议详细阅读，该 PR 展示了如何诊断并修复因非单调 token 曲线导致的错误，以及通过有限扫描（而非二分）选择最优帧的设计模式。核心经验：当领域知识表明曲线非线性时，避免假设单调性，采用穷举可行范围并取最优的策略。

功能与动机

启用视频输入的 GLM4V 多模态启动时，虚拟配置过程变得缓慢且内存占用严重。PR body 指出，由于 _get_max_video_frames 误认为视觉 token 随帧数单调增长，在非单调曲线上提前停止，导致搜索了 249,977 次 token 估算，最终分配了 8,188 MiB 的虚拟视频 numpy 数组。

实现拆解

1. 限制帧搜索范围：将 _get_max_video_frames 中的无限循环改为有限范围 1 至 _MAX_FRAMES_PER_VIDEO (600)，逐帧评估完整范围而不假设单调性，选择产生最多视觉 token 的可行帧数。
2. 使用实际像素限制：移除硬编码的 max_image_pixels，改为调用 _get_video_max_pixels() 从模型配置或用户参数中获取真实像素预算。
3. 独立编码器 CUDA Graph 尺寸：保持编码器 CUDA Graph 容量固定为 600 帧，不再依赖配置候选项，逻辑从 get_max_frames_per_video 中独立。
4. 删除未使用的方法：根据 review 意见删除 get_num_video_tokens 方法，因为其功能已被内联到 _get_max_video_frames 中。
5. 导入优化：将之前内联导入的 ImageProcessingMixin、BaseVideoProcessor 等移到模块顶部，完成代码清理。
6. 测试补充：添加参数化测试 test_get_max_video_frames_matches_glm_resize 验证不同像素和 token 预算下的帧选择，以及 test_encoder_cudagraph_uses_model_video_frame_limit 检测编码器 CUDA Graph 帧限制。

关键文件：

- vllm/model_executor/models/glm4_1v.py（模块 模型层；类别 source；类型 core-logic；符号 get_num_video_tokens, _get_max_video_frames, _get_video_max_pixels）：核心修改文件，修复视频虚拟帧搜索和内存使用，重写 _get_max_video_frames 逻辑，删除冗余方法，导入新模块。

- tests/models/multimodal/processing/test_glm4_1v.py（模块 模型测试；类别 test；类型 test-coverage；符号 test_get_max_video_frames_matches_glm_resize, test_encoder_cudagraph_uses_model_video_frame_limit）：新增参数化测试验证不同像素和 token 预算下的帧选择结果，以及编码器 CUDA Graph 帧限制的独立性。

关键符号：_get_max_video_frames, get_num_video_tokens, _get_video_max_pixels, test_get_max_video_frames_matches_glm_resize, test_encoder_cudagraph_uses_model_video_frame_limit

关键源码片段

vllm/model_executor/models/glm4_1v.py

核心修改文件，修复视频虚拟帧搜索和内存使用，重写 _get_max_video_frames 逻辑，删除冗余方法，导入新模块。

```
# vllm/model_executor/models/glm4_1v.py ( 部分 )

class Glm4vProcessingInfo(BaseProcessingInfo):
    # ... 其他代码 ...

    def _get_max_video_frames(self, max_tokens: int) -> int:
        target_width, target_height = self.get_image_size_with_most_features()
        max_video_pixels = self._get_video_max_pixels() # 从模型配置或用户参数获取实际像素限制
        num_frames_with_most_features = 1
        max_vision_tokens = 0

        # 遍历支持的帧范围 1...600，不再假设 token 单调增长
        for num_frames in range(1, _MAX_FRAMES_PER_VIDEO + 1):
            _, num_vision_tokens = self._get_vision_info(
                image_width=target_width,
                image_height=target_height,
                num_frames=num_frames,
                max_image_pixels=max_video_pixels, # 使用真实像素预算
            )
            if num_vision_tokens <= max_tokens and num_vision_tokens > max_vision_tokens:
                # 选择在预算内且视觉 token 最多的帧数
                max_vision_tokens = num_vision_tokens
                num_frames_with_most_features = num_frames

        return num_frames_with_most_features

    def _get_video_max_pixels(self) -> int:
        # 从 mm_kwargs 或处理器配置获取最大像素数
        mm_kwargs = self.ctx.get_merged_mm_kwargs({})
        if (override_max_pixels := mm_kwargs.get("max_pixels")) is not None:
            return int(override_max_pixels)
        # 否则从处理器配置的 size 推断
        # ... 具体实现见原始文件 ...
```

tests/models/multimodal/processing/test_glm4_1v.py

新增参数化测试验证不同像素和 token 预算下的帧选择结果，以及编码器 CUDA Graph 帧限制的独立性。

```
# tests/models/multimodal/processing/test_glm4_1v.py ( 部分 )

from unittest.mock import Mock
import pytest
from vllm.model_executor.models.glm4_1v import (
    Glm4vForConditionalGeneration,
    Glm4vProcessingInfo,
)

# 参数化测试：不同像素预算和 token 预算下期望的帧数
@pytest.mark.parametrize(
    ("max_video_pixels", "max_tokens", "expected_num_frames"),
    [
        (47_040_000, 124_988, 11),
        (47_040_000, 30_000, 24),
        (100_352_000, 124_988, 21),
        (100_352_000, 30_000, 7),
        (100_352_000, 0, 1),
    ],
)
def test_get_max_video_frames_matches_glm_resize(
    max_video_pixels: int,
    max_tokens: int,
    expected_num_frames: int,
):
    info = Mock(spec=Glm4vProcessingInfo)
    info.get_image_size_with_most_features.return_value = (2184, 2184)
    info._get_video_max_pixels.return_value = max_video_pixels
    vision_config = info.get_hf_config.return_value.vision_config
    vision_config.patch_size = 14
    vision_config.spatial_merge_size = 2
    vision_config.temporal_patch_size = 2
    info._get_vision_info.side_effect = lambda **kwargs: (
        Glm4vProcessingInfo._get_vision_info(info, **kwargs)
    )

    num_frames = Glm4vProcessingInfo._get_max_video_frames(
        info,
        max_tokens=max_tokens,
    )

    assert num_frames == expected_num_frames
    # 验证只调用了一次 _get_video_max_pixels
    assert info._get_video_max_pixels.call_count == 1
```

```
    # 验证恰好执行了 600 次 token 估算（即完整扫描范围）
    assert info._get_vision_info.call_count == 600


def test_encoder_cudagraph_uses_model_video_frame_limit():
    model = Mock()
    # 编码器 CUDA Graph 容量固定为 600 帧，不依赖于 profiling 结果
    assert Glm4vForConditionalGeneration.get_max_frames_per_video(model) == 600
```

评论区精华

Reviewer JaredforReal 在文件 vllm/model_executor/models/glm4_1v.py 第 1182 行提出疑问：get_num_video_tokens 是否还有必要保留？作者回复确认不再需要，并随后在实现中删除了该方法。该讨论已解决，无未决疑问。

- 删除未使用的 get_num_video_tokens 方法 (design): 作者删除该方法，reviewer 无进一步异议。
- Claude bot 自动评论 (other): 无影响。
- PR 总体批准 (other): 批准合并。

风险与影响

- 风险：
  1. 回归风险：变更仅限于 GLM4V 模型的 _get_max_video_frames 及周边方法，对同一代码库中的其他模型无影响。但帧选择逻辑从单调假设改为完整扫描，可能改变极端配置下的帧数选择，测试覆盖了多种像素和 token 组合，风险较低。
  2. 性能风险：从潜在无限循环改为固定 600 次估算，性能可预测且显著更优。但若未来 _MAX_FRAMES_PER_VIDEO 增大，估算次数线性增加，需保持敏感性。
  3. 内存风险：虚拟视频分配从 600 帧全尺寸改为实际估算的少量帧，内存降低约 96.5%，无新的内存风险引入。
  4. 兼容性风险：_get_video_max_pixels 的引入依赖 Hugging Face 处理器配置，若配置缺失或格式变化可能引发异常，但提供了 fallback 行为。 - 影响：影响范围：直接影响 GLM4V（包括 GLM-4.1V 和 GLM-4.6V-Flash）多模态模型的服务启动阶段。影响程度：显著——虚拟视频内存从 8GB 降至约 280MB，启动时的 CPU 和内存压力大幅下降，间接提升整体服务稳定性。不涉及运行时推理路径，因此对输出质量和吞吐量无影响。对团队：维护代码更清晰，删除了冗余方法，导入了必要工具函数。
- 风险标记：核心路径变更，内存使用优化，非单调 token 假设修复

关联脉络

- PR #47155 [GLM4V] Related fix (details not available in this context): PR body 提到该 PR 是验证 #47155 时发现的后续修复，说明二者属于同一功能线的连续改进。