

PR #56721 完整报告

vllm-project/vllm

[Bugfix][Gemma 4] Don't read fft_length when profiling unified audio

合并时间: 2026-09-15 22:57

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56721>

执行摘要

- 一句话: Gemma 4 unified 变体按音频预算推算 dummy 长度, 修启动崩溃
- 推荐动作: 值得精读, 但重点不在 diff 本身 (新增一个 24 行的 helper), 而在两处: 一是 PR 描述对 latent bug 触发条件的完整推导——compute_mm_encoder_budget 的 max() 选择机制、per-item token 对比表 (image 280 / audio 750 / video 2496)、以及 min(t, audio_seq_length) 与 25 token/s 分帧如何把 750 token 绑定到 30.0 s, 这套分析可以复用到其它多模态模型的 dummy 输入与 profiling 问题上; 二是 getattr 探测改为 isinstance 分流的评审交锋, 作者在 commit message 中对「症状 vs 类型区分」的总结值得借鉴。阅读时建议同时关注 PR 描述末尾关于 fft_length 语义的遗留问题, 它是后续可能的独立修复入口。

功能与动机

PR body 明确指出, `Gemma4AudioFeatureExtractor.__init__` 会设置 `self.fft_length`, 而 unified 变体使用的 `Gemma4UnifiedAudioFeatureExtractor` 未定义该属性, 于是 memory profiling 在引擎启动完成前就抛出 `AttributeError: 'Gemma4UnifiedAudioFeatureExtractor' object has no attribute 'fft_length'`。这个 bug 之所以潜伏, 是因为 `compute_mm_encoder_budget` 对 `get_mm_max_tokens_per_item` 取 `max()` 并只 profile 胜出的那个模态: 默认 per-item token 数为 image 280、audio 750 (`audio_seq_length`)、video 2496 (32 帧 x 78), video 胜出, 音频分支根本不执行。但 `_compute_audio_num_tokens` 以 `return min(t, audio_seq_length)` 收尾、mel 分帧约 25 token/s, 所以要把音频上限提到 30 s 以上就必须调大 `audio_seq_length`, 此时 audio 赢得 `max()`, profiler 切到音频分支, 引擎从此无法启动——这正是该修复要覆盖的场景。

实现拆解

变更入口: `vllm/model_executor/models/gemma4_mm.py` 中

`Gemma4DummyInputsBuilder.get_dummy_mm_data` 的音频分支, 原先用 `processor.feature_extractor.fft_length` 作为 `_get_dummy_audios` 的 `length`。

1. 抽出可单测的长度计算函数: 新增模块级 `_dummy_audio_num_samples(processor: Any) -> int`, 把「一条 dummy 音频有多少采样点」从 builder 方法里解耦成纯函数, 使其可以脱离 `transformers.models.gemma4` 的真实处理器被直接断言。
2. 按提取器类型分流: `isinstance(feature_extractor, Gemma4AudioFeatureExtractor)` 为真时返回 `feature_extractor.fft_length`, tower (带 mel 前端) 路径行为不变; 否则读取

`feature_extractor.sampling_rate`, 用 `processor.audio_seq_length`

`*processor.audio_ms_per_token / 1000.0` 换算出模型真正接受的最大音频时长（秒），再乘采样率取整返回。选这两个字段是因为本文件 `_call_hf_processor` 里判断「音频过长」告警用的就是同一对配置，保证 `dummy` 与校验口径一致。

3. 替换调用点与导入：`get_dummy_mm_data` 中 `if num_audios > 0`: 分支改为 `audio_len = _dummy_audio_num_samples(processor)`；文件顶部从 `transformers.models.gemma4` 追加导入 `Gemma4AudioFeatureExtractor`，与既有的 `Gemma4Processor`、`Gemma4Config` 同源。
4. 测试配套：`tests/models/multimodal/processing/test_gemma4_unified.py` 新增两个测试。
`test_dummy_audio_length_falls_back_for_the_unified_extractor` 用桩类 `_UnifiedFeatureExtractor`（只有 `sampling_rate = 16000`）和 `_UnifiedProcessor`（`audio_seq_length = 3000`、`audio_ms_per_token = 40`）断言结果为 `120 * 16000 = 1,920,000`；`test_dummy_audio_length_still_uses_fft_length_for_the_tower_extractor` 用继承 `Gemma4AudioFeatureExtractor` 并绕过真实 `__init__` 的 `_TowerFeatureExtractor` 断言仍返回 `512`，锁定旧行为。
5. 有意未覆盖的部分：`tower` 路径继续把 `fft_length`（FFT 窗口长度，`2 ** ceil(log2(frame_length))`，通常只有几百采样点）当作样本数，作者在 PR 描述的 Note for reviewers 中标记为可疑但刻意不在本 PR 修改，以保证修复范围只限崩溃本身。

关键文件：

- `vllm/model_executor/models/gemma4_mm.py`（模块 多模态模型；类别 `source`；类型 `data-contract`；符号 `_dummy_audio_num_samples`, `Gemma4DummyInputsBuilder.get_dummy_mm_data`, `Gemma4AudioFeatureExtractor`）：核心修复文件：新增 `_dummy_audio_num_samples` 并按提取器类型分流，替换 `get_dummy_mm_data` 中无条件读取的 `processor.feature_extractor.fft_length`，直接决定 `unified` 变体能否完成 `memory profiling` 启动。
- `tests/models/multimodal/processing/test_gemma4_unified.py`（模块 多模态测试；类别 `test`；类型 `test-coverage`；符号 `test_dummy_audio_length_falls_back_for_the_unified_extractor`, `_UnifiedFeatureExtractor`, `_UnifiedProcessor`, `test_dummy_audio_length_still_uses_fft_length_for_the_tower_extractor`）：回归测试：用桩 `processor` 分别模拟 `unified` 与 `tower` 两种提取器形状，同时锁定新的回退公式和 `tower` 路径的旧行为，是本 PR 唯一的自动化验证手段。

关键符号：`_dummy_audio_num_samples`, `Gemma4DummyInputsBuilder.get_dummy_mm_data`, `test_dummy_audio_length_falls_back_for_the_unified_extractor`, `test_dummy_audio_length_still_uses_fft_length_for_the_tower_extractor`

关键源码片段

`vllm/model_executor/models/gemma4_mm.py`

核心修复文件：新增 `_dummy_audio_num_samples` 并按提取器类型分流，替换 `get_dummy_mm_data` 中无条件读取的 `processor.feature_extractor.fft_length`，直接决定 `unified` 变体能否完成 `memory profiling` 启动。

```
# vllm/model_executor/models/gemma4_mm.py
```

```
def _dummy_audio_num_samples(processor: Any) -> int:
```

```
    """Length in samples of a worst-case dummy audio item.
```

```
    `fft_length` 由 `Gemma4AudioFeatureExtractor.__init__` 设置，  
    而 encoder-free 的 Gemma 4 变体使用的 `Gemma4UnifiedAudioFeatureExtractor`  
    并未定义该属性；无条件读取会在该变体的显存 profiling 阶段抛 `AttributeError`，  
    只要 audio 成为 per-item token 数最大的模态，就会命中这条路径。
```

对其它提取器，回退到 processor 自己声明的音频预算 —— 也就是本文件里
用于告警「音频过长」的同一对配置值。

```
    """
```

```
    feature_extractor = processor.feature_extractor
```

```
    # tower（带 mel 前端）路径：保持原行为，直接拿 FFT 窗口长度当样本数
```

```
    if isinstance(feature_extractor, Gemma4AudioFeatureExtractor):
```

```
        return feature_extractor.fft_length
```

```
    # unified（encoder-free）路径：按最大可接受时长反推最坏情况采样点数，
```

```
    # `audio_seq_length` 与 `audio_ms_per_token` 即 `_call_hf_processor` 中
```

```
    # 判断超长音频所用的同一段字段
```

```
    sampling_rate = feature_extractor.sampling_rate
```

```
    max_duration_s = processor.audio_seq_length * processor.audio_ms_per_token / 1000.0
```

```
    return int(max_duration_s * sampling_rate)
```

```
# ... 同一文件内，`Gemma4DummyInputsBuilder.get_dummy_mm_data` 的调用点：
```

```
class Gemma4DummyInputsBuilder(BaseDummyInputsBuilder[Gemma4ProcessingInfo]):
```

```
    def get_dummy_mm_data(self, seq_len, mm_counts, mm_options=None):
```

```
        # ...（省略 image / video 分支与 override 校验）
```

```
        if num_audios > 0:
```

```
            # 不再直接访问 `processor.feature_extractor.fft_length`，
```

```
            # 统一走 helper，避免 unified 变体在启动期 profiling 崩溃
```

```
            audio_len = _dummy_audio_num_samples(processor)
```

```
            data["audio"] = self._get_dummy_audios(
```

```
                length=audio_len,
```

```
                num_audios=num_audios,
```

```
                overrides=audio_overrides,
```

```
            )
```

```
        return data
```

tests/models/multimodal/processing/test_gemma4_unified.py

回归测试：用桩 processor 分别模拟 unified 与 tower 两种提取器形状，同时锁定新的回退公式和 tower 路径的旧行为，是本 PR 唯一的自动化验证手段。

```
# tests/models/multimodal/processing/test_gemma4_unified.py
```

```
def test_dummy_audio_length_falls_back_for_the_unified_extractor():
```

```
    """unified 音频提取器不定义 `fft_length`。
```

旧实现无条件读取该属性，导致所有「audio 是 per-item token 数最大模态」的 unified Gemma 4 部署在显存 profiling 阶段抛 `AttributeError`，而只要把 `audio\_seq\_length` 抬到 video 预算之上就会进入该情形。

```
class _UnifiedFeatureExtractor:
    # unified 变体只有采样率，没有 `fft_length`
    sampling_rate = 16000

class _UnifiedProcessor:
    feature_extractor = _UnifiedFeatureExtractor()
    audio_seq_length = 3000
    audio_ms_per_token = 40

# 3000 token * 40 ms = 120 s, 16 kHz 下即 1,920,000 个采样点
assert _dummy_audio_num_samples(_UnifiedProcessor()) == 120 * 16000

def test_dummy_audio_length_still_uses_fft_length_for_the_tower_extractor():
    """mel/tower 提取器保持原有行为。"""

    class _TowerFeatureExtractor(Gemma4AudioFeatureExtractor):
        def __init__(self): # noqa: D107 - 绕过真实特征提取器的初始化
            self.fft_length = 512
            self.sampling_rate = 16000

    class _TowerProcessor:
        feature_extractor = _TowerFeatureExtractor()
        audio_seq_length = 750
        audio_ms_per_token = 40

    # 走 `isinstance` 分支，返回值与修复前逐字一致
    assert _dummy_audio_num_samples(_TowerProcessor()) == 512
```

评论区精华

评审只有一条代码级评论，但指向了实现的关键表达方式：DarkLight1337 在 [vllm/model_executor/models/gemma4_mm.py](#) 上指出 "Perhaps it's better to use `isinstance` checks explicitly"。作者随后在 commit 8ba5f21 中把初版的 `getattr(processor.feature_extractor, "fft_length", None)` 探测改写为显式 `isinstance` 判断，并在 commit message 里总结得很清楚："The `getattr` probe described the symptom -- 'this object happens to have `fft_length`' -- rather than the distinction that actually matters, which is which feature extractor the variant is using."。

第二个讨论点是 PR 描述末尾的遗留问题：把 `fft_length` 当样本数在 tower 路径上也不合理，会让 dummy 音频远短于模型可接受的音频、profiling 预留偏小。DarkLight1337 在 issue 评论里专门 @Isotr0py: "could you take a look as well? Especially that last comment in the PR description"，但这条线索在本次讨论中没有得出「是否值得跟进」的结论。

第三点是验证状态：作者在 Test Result 主动更正了早前版本 " 测试通过 " 的表述，说明环境缺少 `transformers.models.gemma4`、两个新测试未本地执行，CI 也因 `pre-run-check` 门控未跑；随后 Isotr0py 评论 `/ci run` 触发 Buildkite CI #89090 (commit a4fda6c2)，PR 最终被 DarkLight1337 与 Isotr0py 双双 approve 并合并。

- 用 `isinstance` 显式判断替代 `getattr` 属性探测 (design): 作者在 commit 8ba5f21 改为 `isinstance(feature_extractor, Gemma4AudioFeatureExtractor)`，并从 `transformers.models.gemma4` 显式导入该类，同时把理由写进 commit message: `getattr` 探测描述的是症状（这对象恰好有 `fft_length`），而真正要区分的是变体在用哪种特征提取器。
- tower 路径把 `fft_length` 当 dummy 采样数是否合理 (design): 作者明确表示有意不在本 PR 修改，以把范围限制在崩溃修复上，并愿意另行跟进；评论串中未给出是否值得跟进或如何修正的结论。
- 新测试未在本地执行、CI 需 maintainer 手动触发 (testing): Isotr0py 评论 `/ci run` 触发 Buildkite CI #89090 (commit a4fda6c2)，DarkLight1337 与 Isotr0py 均 approve 并合并；材料中未展示 CI 最终结果。

风险与影响

- 风险：
 1. 回退分支的样本量可能显著放大 profiling 开销: unified 路径按最坏情况推算，示例中 $3000 \text{ token} \times 40 \text{ ms} = 120 \text{ s}$ ，在 16 kHz 下得到 1,920,000 采样点（float32 约 7.7 MB/ 条），远大于原先 `fft_length` 的几百采样点；当 `limit_mm_per_prompt` 中 `audio` 数量较大时，memory profiling 的内存与耗时都会上升。这是「修正正确性换启动成本」的取舍，但 PR 未给出量级评估。
 2. 依赖处理器字段的隐式契约: 该分支假设 `processor.audio_seq_length`、`processor.audio_ms_per_token`、`feature_extractor.sampling_rate` 三者在所有非 tower 提取器上都存在且与真实分帧逻辑一致；若未来出现第三种提取器或采样率与实际重采样不符，dummy 长度会与实际处理路径偏离（`getattr` 改为 `isinstance` 后，非 `Gemma4AudioFeatureExtractor` 但缺字段的情况会直接 `AttributeError`，只是报错位置从 `fft_length` 换到了别的字段）。
 3. 新增测试未经本地执行: 作者明确说明因环境缺少 `transformers.models.gemma4` 而未能运行，且 CI 靠 maintainer 手动触发；若测试模块的模块级导入或断言在目标环境不成立，这次回归保护的有效性存疑（最终 CI 已触发，但材料未展示结果）。
 4. 遗留的 profiling 低估: tower 路径仍以 `fft_length` 作为 dummy 样本数，使 `encoder/activation` 的峰值显存被低估，属于本 PR 之外但已被点名的隐患。
 5. 兼容性: `Gemma4AudioFeatureExtractor` 从 `transformers.models.gemma4` 顶层导入，需要 `transformers` 版本提供该类；该文件已依赖同模块的 `Gemma4Config/Gemma4Processor`，因此实际风险有限，但属于新增的模块级导入依赖。
 - 影响: 用户侧: 默认配置（video 2496 token 胜出 profile）下行为与之前完全一致，没有回归；受影响的是把 `audio_seq_length` 调到超过 video 预算、用 unified 变体服务 30 s 以上音频的部署，这些部署从「启动即 `AttributeError` 崩溃」恢复为可正常启动。tower 变体走 `isinstance` 分支，返回值与旧实现逐字相同。系统侧: unified 变体的

memory profiling 阶段会构造更长的 dummy 音频（按最坏音频预算），启动阶段的内存占用与耗时上升，但换来的是 profiler 不再依赖一个在 encoder-free 变体上不存在的属性。团队侧：该 PR 为多模态 processor 的「变体分支判断」提供了一个可复用范式——按类类型 isinstance 判断分支，而不是用 getattr 探测属性是否存在；这条评审意见来自核心 maintainer，值得作为后续同类修复的默认写法。

- 风险标记：启动期 profiling 成本上升，回退分支依赖隐式字段契约，新测试未在作者环境执行，tower 路径 profiling 低估遗留，新增 transformers 模块级导入依赖

关联脉络

- PR #53124 [Bugfix] Load reasoning parser plugins before headless engine config: 同属「仅在特定配置下触发的启动期崩溃」类修复：都是把隐式前提显式化（那边是提前导入插件，这边是按提取器类型分流），可对照两者的根因分析方法。
- PR #56908 [Model Runner V2] Tolerate lack of pinned memory support: 同属「为目标平台 / 变体缺失能力补回退分支」的模式：MRV2 在无 pinned memory 平台走回退缓冲，本 PR 为没有 fft_length 的 unified 提取器回退到音频预算推算。
- PR #56969 [Bugfix] Prevent out-of-bounds access in FlashInfer SM90 sparse MLA mixed batches: 同为变体 / 特殊批次组合下才会暴露的边界崩溃修复，且都带小范围回归测试，属于同一时期仓库内的同类质量修复脉络。