PR #45478 完整报告

vllm-project/vllm

[CI Bug] Fix `ValueError: There is no module or parameter named 'model.vision_tower.vision_model'`

合并时间：2026-06-13 17:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45478

执行摘要

- 一句话：修复多模态模型权重映射时丢失 WeightRenaming 的 bug
- 推荐动作：建议合入并尽快回归测试多模态模型。这是一个值得精读的小改动，展示了如何正确处理上游库的权重重命名契约。

功能与动机

修复 Buildkite CI 中 gemma3 多模态模型加载失败的问题（"ValueError: There is no module or parameter named 'model.vision_tower.vision_model' in TransformersMultiModalForCausalLM"）。根本原因是当 Transformers 发布更新并重命名权重时，vLLM 仅存储了编译后的正则表达式，丢失了 WeightRenaming 对象中的完整重命名逻辑。

实现拆解

1. 在 WeightsMapper 类中新增字段（vllm/model_executor/models/utils.py）：添加 orig_to_new_renamings: list[Any] 字段，用于存储原始的 WeightRenaming 对象列表。同时更新 __or__ 方法以合并两个 WeightsMapper 实例中的该列表。
2. 在 _map_name 方法中优先应用原始重命名（vllm/model_executor/models/utils.py）：在 _map_name 方法开头，先遍历 orig_to_new_renamings，对每个 WeightRenaming 调用 renaming.rename_source_key(key)，更新键名。之后再应用原有的正则、子串、前缀、后缀映射。这确保了 Transformers 官方提供的重命名逻辑优先执行。
3. 简化 _create_hf_to_vllm_mapper 中的代码（vllm/model_executor/models/transformers/base.py）：将原来将 WeightRenaming 编译为正则并存入 orig_to_new_regex 的代码，替换为直接将 WeightRenaming 对象追加到 orig_to_new_renamings 列表。去除了 re.compile 和 target_pattern 的提取，逻辑更加清晰且正确。

关键文件：

- vllm/model_executor/models/utils.py（模块 模型加载器；类别 source；类型 data-contract；符号 WeightsMapper, WeightsMapper.or, WeightsMapper._map_name）：核心变更：在 WeightsMapper 中新增 orig_to_new_renamings 字段，并在 _map_name 中优先调用 rename_source_key，解决了权重重命名丢失的问题。

- vllm/model_executor/models/transformers/base.py（模块 模型基类；类别 source；类型 data-contract；符号 TransformersMultiModalForCausalLM._create_hf_to_vllm_mapper）：配合变更：将原来把 WeightRenaming 编译为正则并存入 orig_to_new_regex 的逻辑，改为直接存入 orig_to_new_renamings，简化代码并修复 bug。

关键符号：WeightsMapper.init, WeightsMapper.or, WeightsMapper._map_name, TransformersMultiModalForCausalLM._create_hf_to_vllm_mapper

关键源码片段

vllm/model_executor/models/utils.py

核心变更：在 WeightsMapper 中新增 orig_to_new_renamings 字段，并在 _map_name 中优先调用 rename_source_key，解决了权重重命名丢失的问题。

```
@dataclass
class WeightsMapper:
    """Maps the name of each weight if they match the following patterns.

    If a key maps to a value of `None`, the corresponding weight is ignored."""

    # 新增字段：存储原始的 WeightRenaming 对象，保留 Transformers 提供的完整重命名逻辑
    orig_to_new_renamings: list[Any] = field(default_factory=list)
    orig_to_new_regex: Mapping[re.Pattern, str | None] = field(default_factory=dict)
    orig_to_new_substr: Mapping[str, str | None] = field(default_factory=dict)
    orig_to_new_prefix: Mapping[str, str | None] = field(default_factory=dict)
    orig_to_new_suffix: Mapping[str, str | None] = field(default_factory=dict)

    def __or__(self, other: "WeightsMapper") -> "WeightsMapper":
        """Combine two `WeightsMapper`s by merging their mappings."""
        return WeightsMapper(
            # 合并时同时合并 orig_to_new_renamings 列表
            orig_to_new_renamings=[
                *self.orig_to_new_renamings,
                *other.orig_to_new_renamings,
            ],
            orig_to_new_regex={**self.orig_to_new_regex, **other.orig_to_new_regex},
            orig_to_new_substr={**self.orig_to_new_substr, **other.orig_to_new_substr},
            orig_to_new_prefix={**self.orig_to_new_prefix, **other.orig_to_new_prefix},
            orig_to_new_suffix={**self.orig_to_new_suffix, **other.orig_to_new_suffix},
        )

    def _map_name(self, key: str) -> str | None:
        # 优先应用原始 WeightRenaming 的重命名逻辑
        for renaming in self.orig_to_new_renamings:
            key, _ = renaming.rename_source_key(key)

        for pattern, new_key in self.orig_to_new_regex.items():
            if pattern.search(key):
                if new_key is None:
```

```
                return None
            key = pattern.sub(new_key, key)

        for substr, new_key in self.orig_to_new_substr.items():
            if substr in key:
                if new_key is None:
                    return None
                key = key.replace(substr, new_key, 1)

        for prefix, new_key in self.orig_to_new_prefix.items():
            if key.startswith(prefix):
                if new_key is None:
                    return None
                key = key.replace(prefix, new_key, 1)

        for suffix, new_key in self.orig_to_new_suffix.items():
            if key.endswith(suffix):
                if new_key is None:
                    return None
                key = new_key.join(key.rsplit(suffix, 1))

        return key
```

vllm/model_executor/models/transformers/base.py

配合变更：将原来把 WeightRenaming 编译为正则并存入 orig_to_new_regex 的逻辑，改为直接存入 orig_to_new_renamings，简化代码并修复 bug。

```
def _create_hf_to_vllm_mapper(self):
    self.hf_to_vllm_mapper = WeightsMapper()
    orig_to_new_renamings = self.hf_to_vllm_mapper.orig_to_new_renamings
    orig_to_new_regex = self.hf_to_vllm_mapper.orig_to_new_regex

    for mapping in get_model_conversion_mapping(self.model):
        if isinstance(mapping, WeightRenaming):
            # 直接存储 WeightRenaming 对象，保留完整的重命名逻辑
            orig_to_new_renamings.append(mapping)
        # TODO: Handle WeightConverter to enable layer merging

    # 后续处理 unexpected weights, base model prefix 等（与之前相同）
    # ...
```

评论区精华

无 review 评论讨论。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更只影响权重名称映射路径，且保留了原有的正则映射作为后续处理。但如果其他代码依赖 orig_to_new_regex 中的 WeightRenaming 衍生正则，可能会受影响，不过经分析 _create_hf_to_vllm_mapper 是唯一写入点，且新增的 orig_to_new_renamings 在 _map_name 中优先执行，不影响后续正则匹配。
- 影响：修复了多模态模型（尤其是 gemma3 等使用 Transformers 较新版本权重重命名的模型）在加载时因权重名称映射错误而崩溃的问题。影响范围限于使用 WeightsMapper 的模型加载流程。对于不涉及 WeightRenaming 的模型（如 LLaMA），无行为变化。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- 暂无明显关联 PR