PR #44450 完整报告

vllm-project/vllm

[Model Runner V2] Fix mrv2 mm lora issue

合并时间：2026-06-09 02:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44450

执行摘要

- 一句话：修复 V2 模型运行器多模态 LoRA 映射未设置导致测试失败
- 推荐动作：此 PR 修复明确，测试充分，建议立即合并。值得关注的设计是将 LoRA 激活与多模态 embedding 生成解耦到独立模块，未来维护时可借鉴。

功能与动机

在使用 V2 模型运行器时，tests/lora/test_qwenvl.py::test_qwen3vl_vision_lora 测试因生成结果不匹配而失败，原因是多模态 embedding 生成前缺少 LoRA 激活步骤。PR body 中贴出了具体的 AssertionError。

实现拆解

1. 新增 vllm/v1/worker/gpu/mm/lora.py：定义 set_active_mm_loras 函数，遍历 scheduled_encoder_inputs，为每个请求计算 LoRA 映射，并分两次（TOWER 和 CONNECTOR）调用 lora_manager.set_active_adapters。
2. 修改 vllm/v1/worker/gpu/model_runner.py：在 GPUModelRunner.execute_model 中，当 lora_config 不为 None 时，在调用 get_mm_embeddings 之前插入 set_active_mm_loras 调用。
3. 补充测试 tests/v1/worker/test_gpu_model_runner.py：新增 test_set_active_mm_loras_builds_tower_and_connector_mappings，通过 Mock 对象验证 TOWER 和 CONNECTOR 映射的正确性，包括 prompt_mapping、index_mapping 和调用次数。

关键文件：

- vllm/v1/worker/gpu/mm/lora.py（模块 多模态 LoRA；类别 source；类型 core-logic；符号 set_active_mm_loras）：新增核心函数 set_active_mm_loras，实现多模态 LoRA 映射构建与激活逻辑，是本 PR 核心变更。
- tests/v1/worker/test_gpu_model_runner.py（模块 测试；类别 test；类型 test-coverage；符号 test_set_active_mm_loras_builds_tower_and_connector_mappings）：新增单元测试覆盖 set_active_mm_loras 的 TOWER 和 CONNECTOR 映射构建逻辑，是质量保证的关键。
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 data-contract）：在 execute_model 中集成 set_active_mm_loras 的调用点，确保多模态

embedding 生成前激活 LoRA。

关键符号：set_active_mm_loras, test_set_active_mm_loras_builds_tower_and_connector_mappings

关键源码片段

tests/v1/worker/test_gpu_model_runner.py

新增单元测试覆盖 set_active_mm_loras 的 TOWER 和 CONNECTOR 映射构建逻辑，是质量保证的关键。

```
def test_set_active_mm_loras_builds_tower_and_connector_mappings():
    # Mock 模型：定义 encoder 和 connector token 数的计算方式
    model = Mock()
    model.get_num_mm_encoder_tokens.side_effect = lambda num_embeds: num_embeds + 1
    model.get_mm_mapping.return_value = SimpleNamespace(connector=True)
    model.get_num_mm_connector_tokens.side_effect = lambda num_tokens: num_tokens + 10

    # Mock LoRA 管理器，声明支持 tower/connector LoRA
    lora_manager = Mock()
    lora_manager.supports_tower_connector_lora.return_value = True

    # 构建 encoder cache：包含两个请求的两个图像特征
    encoder_cache = EncoderCache()
    encoder_cache.mm_features["req-with-lora"] = [
        MultiModalFeatureSpec(
            data=None,
            modality="image",
            identifier="img-0",
            mm_position=PlaceholderRange(offset=0, length=2),
        ),
        MultiModalFeatureSpec(
            data=None,
            modality="image",
            identifier="img-1",
            mm_position=PlaceholderRange(offset=2, length=3),
        ),
    ]
    encoder_cache.mm_features["req-no-lora"] = [
        MultiModalFeatureSpec(
            data=None,
            modality="image",
            identifier="img-2",
            mm_position=PlaceholderRange(offset=0, length=1),
        )
    ]

    # 构造 LoraState：一个请求带 LoRA（ID 7），另一个不带（ID 0）
    lora_state = LoraState(max_num_reqs=4)
```

```
lora_request = LoRARequest("vision-lora", 7, "/tmp/vision-lora")
lora_state.add_request("req-with-lora", 0, lora_request)
lora_state.add_request("req-no-lora", 1, None)

# 调用被测函数
set_active_mm_loras(
    model=model,
    lora_manager=lora_manager,
    encoder_cache=encoder_cache,
    req_id_to_index={
        "req-with-lora": 0,
        "req-no-lora": 1,
    },
    lora_state=lora_state,
    scheduled_encoder_inputs={
        "req-with-lora": [1, 0], # 注意：先取 idx=1 再 idx=0（顺序不重要但会影响计数）
        "req-no-lora": [0],
        "missing-req": [0], # 此请求不在 req_id_to_index 中，会被跳过
    },
)

# 验证 set_active_adapters 被调用了两次（TOWER 和 CONNECTOR）
assert lora_manager.set_active_adapters.call_count == 2

# 检查第一次调用（TOWER 类型）
tower_requests, tower_mapping = lora_manager.set_active_adapters.call_args_list[0].args
assert tower_requests == {lora_request}
assert tower_mapping.type is LoRAMappingType.TOWER
# 映射验证：req-with-lora (id=7) 的两个图像分别有 3 和 4 个 token（num_embeds+1），
# req-no-lora (id=0) 的一个图像有 2 个 token，所以 prompt_mapping 应为 (7,7,0)
assert tower_mapping.prompt_mapping == (7, 7, 0)
# token_mapping 应为 7 重复 3 次 + 7 重复 4 次 + 0 重复 2 次
assert tower_mapping.index_mapping == (7,7,7,7,7,7,7,0,0)

# 检查第二次调用（CONNECTOR 类型）
conn_requests, conn_mapping = lora_manager.set_active_adapters.call_args_list[1].args
assert conn_requests == {lora_request}
assert conn_mapping.type is LoRAMappingType.CONNECTOR
# connector token 数为 encoder token 数加 10，所以 index_mapping 长度分别为 13+14+12?
# 但按代码逻辑：connector_token_mapping 由 np.repeat 生成，即每个 prompt_lora_id
重复其对应的 connector_token_count 次
# 对于 7,7,0 分别重复 (3+10=13), (4+10=14), (2+10=12) 总共 13+14+12=39
assert len(conn_mapping.index_mapping) == 39
assert all(x == 7 for x in conn_mapping.index_mapping[:13])
assert all(x == 7 for x in conn_mapping.index_mapping[13:27])
assert all(x == 0 for x in conn_mapping.index_mapping[27:])
assert conn_mapping.prompt_mapping == tower_mapping.prompt_mapping
```

评论区精华

Reviewer jeejeelee 要求增加单元测试，作者 yewentao256 随后提交了测试用例（commit add test），审阅通过。

- 缺少单元测试覆盖 (testing): 作者 yewentao256 随后提交了测试用例（commit add test），审阅者批准。

风险与影响

- 风险：主要风险在于新逻辑在 execute_model 中每个步骤前置调用，可能对非 LoRA 场景有额外性能开销（但通过 if self.lora_config is not None 条件避免）。新增函数依赖 model.get_num_mm_encoder_tokens 等接口，若模型未实现则返回不会生效。测试覆盖了两类请求场景，但仍需关注其他多模态模型的兼容性。
- 影响：直接影响使用 V2 模型运行器且启用 LoRA 的多模态推理用户。修复了 Qwen2VL 等模型的 LoRA 推理错误。对其他不使用 V2 模型运行器或无 LoRA 的场景无影响。
- 风险标记：核心路径变更，缺少回归测试覆盖（test 已补）

关联脉络

- 暂无明显关联 PR