

PR #55548 完整报告

vllm-project/vllm

[Bugfix][LoRA] Use stored rsLoRA scaling factor in MoE expert packing

合并时间: 2026-09-11 11:52

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55548>

执行摘要

- 一句话: 修复 MoE LoRA 打包忽略 rsLoRA 缩放因子
- 推荐动作: 值得精读。这是一个典型的“单一事实来源”修复: 优先使用 PEFTHelper 已存储的 per-adapter 缩放因子, 而非在打包路径中重新推导, 避免语义漂移。测试设计 (参数化两条路径 + 保留非 gated 特例) 也值得借鉴, 且删除了无回归信号的低价值参数化。

功能与动机

issue #55545 报告: 使用 rank-stabilized LoRA (`use_rslora: true`) 训练的适配器在作用于 MoE 专家模块时被应用了错误的缩放量级。PEFTHelper 为每个适配器计算并存储 $vllm_lora_scaling_factor = lora_alpha / \sqrt{r}$, dense 路径通过 `pack()+optimize()` 正确使用, 但 `pack_moe` 与 `pack_moe_stacked` 重新计算为 $lora_alpha / rank$, 导致 $r=16$ 、 $lora_alpha=32$ 时正确因子 8 被应用成 2, 适配器 delta 静默弱化 $\sqrt{r}$ 倍。

实现拆解

1. 修改 `pack_moe` (`vllm/lora/lora_weights.py`): 将 `scaling = lora_alpha / rank` 替换为使用该组专家中第一个 (代表性) LoRA 上已存储的 `first_lora.scaling`; 保留 `is_non_gated_moe` 分支中 `last_scaling = 1.0` 的逻辑, 防止 w_3 复用 w_1 权重时二次缩放。
2. 修改 `pack_moe_stacked`: 同样将重算替换为 `w1_lora.scaling`, $w_1/w_2/w_3$ 统一使用存储因子。
3. 新增 `tests/lora/test_lora_weights.py`: 参数化覆盖 `pack_moe` 与 `pack_moe_stacked`, 断言打包后 `scaling` 仍为 $alpha / \sqrt{r}$; 另测非 gated MoE 时 w_3 `scaling` 为 1.0。
`autouse fixture should_do_global_cleanup_after_test` 返回 `False`, 跳过 GPU/ray 清理, 保持纯 CPU 单元测试。
4. 无配置 `/schema/` 部署改动; CI 由 jeejeelee 触发两次 (Buildkite #87945、#88233)。

关键文件:

- `vllm/lora/lora_weights.py` (模块 LoRA 权重; 类别 source; 类型 core-logic; 符号 `pack_moe`, `pack_moe_stacked`): 核心修复所在: 两条 MoE 打包路径的缩放因子来源从重算改为使用已存储值, 直接决定 rsLoRA 适配器在 MoE 上的数值正确性。
- `tests/lora/test_lora_weights.py` (模块 LoRA 测试; 类别 test; 类型 test-coverage; 符号 `should_do_global_cleanup_after_test`, `moe_loras`, `test_moe_packing_uses_stored_scaling`, `test_non_gated_moe_keeps_w3_unscaled`):

新增回归测试，覆盖两条打包路径的 rsLoRA 缩放保留和非 gated MoE 的 w3 特例，是修复正确性的关键验证。

关键符号：pack_moe, pack_moe_stacked, test_moe_packing_uses_stored_scaling, test_non_gated_moe_keeps_w3_unscaled

关键源码片段

vllm/lora/lora_weights.py

核心修复所在：两条 MoE 打包路径的缩放因子来源从重算改为使用已存储值，直接决定 rsLoRA 适配器在 MoE 上的数值正确性。

```
# vllm/lora/lora_weights.py (head 版本, pack_moe_stacked 完整实现)
@classmethod
def pack_moe_stacked(
    cls,
    loras: GenericSequence["LoRALayerWeights | None"],
    module_name: str,
) -> "PackedLoRALayerWeights":
    """打包预堆叠 (3D) 的 w1/w2/w3 专家 LoRA, 供 FusedMoEWithLoRA 使用。"""
    assert len(loras) == 3, (
        "shared-outer expert LoRA expects exactly w1/w2/w3 stacked tensors"
    )
    w1_lora, w2_lora, w3_lora = loras
    assert w1_lora is not None and w2_lora is not None and w3_lora is not None
    rank = w1_lora.rank
    lora_alpha = w1_lora.lora_alpha
    # 关键修复：直接使用 LoRALayerWeights 上已存储的 per-adapter scaling。
    # 该因子由 PEFTHelper 按 use_rslora 预先算好 (lora_alpha / sqrt(r)) ,
    # 此处不能再以 lora_alpha / rank 重算，否则 rsLoRA 语义会被静默丢弃。
    scaling = w1_lora.scaling
    return cls(
        module_name,
        rank,
        [lora_alpha, lora_alpha, lora_alpha],
        [w1_lora.lora_a, w2_lora.lora_a, w3_lora.lora_a],
        [w1_lora.lora_b, w2_lora.lora_b, w3_lora.lora_b],
        scaling=[scaling, scaling, scaling],
    )
```

tests/lora/test_lora_weights.py

新增回归测试，覆盖两条打包路径的 rsLoRA 缩放保留和非 gated MoE 的 w3 特例，是修复正确性的关键验证。

```
# tests/lora/test_lora_weights.py (新增, 核心回归测试)
RANK, ALPHA = 16, 32
# PEFTHelper 对 use_rslora 适配器存储 alpha / sqrt(r)
RSLORA_SCALING = ALPHA / RANK**0.5

def moe_loras(scaling, stacked=False):
```

```

# 构造 3 个专家 LoRA，支持 2D（逐专家）或 3D（预堆叠）形态
a_shape = (3, RANK, 4) if stacked else (RANK, 4)
b_shape = (3, 6, RANK) if stacked else (6, RANK)
return [
    LoRALayerWeights(
        module_name=f"experts.w{ij}",
        rank=RANK,
        lora_alpha=ALPHA,
        lora_a=torch.randn(*a_shape),
        lora_b=torch.randn(*b_shape),
        scaling=scaling,
    )
    for i in (1, 2, 3)
]

@pytest.mark.parametrize(
    "packer,stacked",
    [
        (PackedLoRALayerWeights.pack_moe, False),
        (PackedLoRALayerWeights.pack_moe_stacked, True),
    ],
)
def test_moe_packing_uses_stored_scaling(packer, stacked):
    # 打包后 scaling 必须保留 rsLoRA 因子，而不是退化为 alpha / rank
    packed = packer(moe_loras(RSLORA_SCALING, stacked), "experts")
    assert packed.scaling == pytest.approx([RSLORA_SCALING] * 3)

def test_non_gated_moe_keeps_w3_unscaled():
    # 非 gated MoE 将 w1 权重复用为 w3，w3 缩放必须为 1.0，避免双重缩放
    packed = PackedLoRALayerWeights.pack_moe(
        moe_loras(RSLORA_SCALING), "experts", is_non_gated_moe=True
    )
    assert packed.scaling[:2] == pytest.approx([RSLORA_SCALING] * 2)
    assert packed.scaling[2] == 1.0

```

评论区精华

核心讨论只有一条：评审者 jeejeelee 在新增测试文件第 1 行评论“Can we simplify this script?”，作者 kushaldabbe 响应并简化：将两条打包路径合并为单一参数化测试，并删除了普通 scaling 参数化——因为非 rsLoRA 时存储因子与重算因子恒等，无回归信号。最终 jeejeelee 两次触发 CI，linitra24 与 jeejeelee 均批准，无未解决疑虑。

- 简化测试脚本 (testing): 测试脚本按建议简化，保留 rsLoRA 回归用例与非 gated MoE 的 w3 特例。

风险与影响

- 风险：数值风险：修复会改变 rsLoRA+MoE 用户得到的前向输出量级，但属于纠正静默错误；普通 LoRA 的存储因子与 alpha/rank 相等，行为不变。pack_moe 中 first_lora.scaling

的选取逻辑未在 patch 中完整展示，混合有 / 无 LoRA 专家的场景下需确认取到的是正确 expert 组的因子，现有断言 (`w1_lora is not None` 等) 提供基本保护。测试仅 CPU 单元级，未覆盖端到端模型推理和混合专家缺失 LoRA 的边界。无性能、安全、兼容性影响。

- 影响：影响用户：使用 rsLoRA 训练的 LoRA 适配器且目标为 MoE 专家模块（如 Mixtral 等）的用户，修复后适配器强度恢复正确，生成质量会发生变化（变强 $\sqrt{r}$ 倍），这是预期修正。系统影响极小：源码仅 7 行变更，无 API/ 配置变化。团队影响：LoRA 模块维护者需合入，CI 已跑通。
- 风险标记：核心数值修复，单元测试仅 CPU, `first_lora` 取值需确认

关联脉络

- 暂无明显关联 PR