PR #6896 完整报告

verl-project/verl

[fsdp] fix: Fix Qwen3 MoE FSDP weight sync for vLLM rollout in Transformers 5

合并时间：2026-07-03 00:02

原文链接：http://prhub.com.cn/verl-project/verl/pull/6896

执行摘要

- 一句话：修复 Qwen3 MoE FSDP-vLLM 权重同步兼容性
- 推荐动作：建议合并并同步考虑添加单元测试（至少验证 _iter_vllm_compatible_moe_params 的输入输出等价性），以及添加后端配置选项让用户可以手动覆盖版本阈值。

功能与动机

Transformers 5 在 Qwen-style MoE 模型中采用了打包 3D 参数存储（mlp.experts.gate_up_proj），但 vLLM（<=0.24.0）在 FSDP 实时权重同步时仍期望传统的 per-expert 键格式，导致权重加载失败。PR body 明确指出需要扩展 packed 张量以兼容 vLLM 加载路径，且社区确认该问题在 vLLM 0.24.0 之后才被上游修复（vLLM PR#47058）。

实现拆解

该变更仅修改 verl/workers/rollout/vllm_rollout/vllm_rollout.py 一个文件，分三步实现：

1. 版本检测函数_should_expand_vllm_moe_params()：通过解析 vLLM 版本号，决定是否启用 MoE 参数扩展（仅对 vLLM <= 0.24.0 生效），为后续条件性逻辑提供守卫。
2. 异步迭代器_iter_vllm_compatible_moe_params(weights)：接收权重生成器，遍历每个权重项。若发现以 .mlp.experts.gate_up_proj 结尾且为 3D 张量，则将其沿 dim=1 切分为 gate 和 up，再按 expert_id 遍历生成 gate_proj.weight 和 up_proj.weight；对 .mlp.experts.down_proj 类似处理；否则原样透传。该函数保持流式处理，不构造完整字典。
3. 集成到 update_weights 方法：在调用 send_weights 之前，检查是否需扩展且非 PEFT 已完成的基础同步，若满足则将 weights 替换为扩展后的异步迭代器。 未新增测试文件，但已有 reviewer 强调关键 suffix 匹配需含 .weight，实际提交已按 review 建议调整（.gate_up_proj 改为 .gate_up_proj.weight）。

关键文件：

- verl/workers/rollout/vllm_rollout/vllm_rollout.py（模块 推理引擎；类别 source；类型 dependency-wiring；符号 _should_expand_vllm_moe_params, _iter_vllm_compatible_moe_params）：核心变更文件：新增版本检测和 MoE 参数展开逻辑，影响 Qwen3 MoE 模型在 vLLM <=0.24.0 下的权重同步正确性。

关键符号：_should_expand_vllm_moe_params, _iter_vllm_compatible_moe_params

关键源码片段

verl/workers/rollout/vllm_rollout/vllm_rollout.py

核心变更文件：新增版本检测和 MoE 参数展开逻辑，影响 Qwen3 MoE 模型在 vLLM <=0.24.0 下的权重同步正确性。

```
# verl/workers/rollout/vllm_rollout/vllm_rollout.py
from verl.third_party.vllm import get_version
from packaging import version as vs

# 检测 vLLM 版本是否 <= 0.24.0，若低于此版本则需要展开 packed MoE 参数
# 因为 vLLM 直到 0.24.0 才从 upstream 合入 Transformers 5 格式兼容性修复
def _should_expand_vllm_moe_params() -> bool:
    current_version = get_version("vllm")
    if not current_version:
        return False
    try:
        return vs.parse(current_version) <= vs.parse("0.24.0")
    except vs.InvalidVersion:
        return False


async def _iter_vllm_compatible_moe_params(weights):
    """
    将 Transformers 5 打包的 MoE 专家权重（3D 张量）实时展开为 vLLM 期望的 per-expert 键格式。

    Transformers 5 存储格式（Qwen-style MoE）：
        mlp.experts.gate_up_proj.weight: [num_experts, 2*intermediate_size, hidden_size]
        mlp.experts.down_proj.weight: [num_experts, hidden_size, intermediate_size]

    vLLM 期望的格式（per-expert）：
        mlp.experts.{expert_id}.gate_proj.weight
        mlp.experts.{expert_id}.up_proj.weight
        mlp.experts.{expert_id}.down_proj.weight

    此函数保持异步流式生成，避免构造完整字典，降低内存开销。
    """
    # 延迟导入，避免跨文件循环依赖
    from verl.workers.rollout.utils import ensure_async_iterator

    async for name, tensor in ensure_async_iterator(weights):
        # 处理 gate_up_proj: 沿 dim=1 切分为 gate_proj 和 up_proj
        if name.endswith(".mlp.experts.gate_up_proj.weight") and tensor.dim() == 3:
            gate, up = tensor.chunk(2, dim=1)
            base = name.removesuffix(".gate_up_proj.weight")
            for expert_id in range(tensor.size(0)):
                yield f"{base}.{expert_id}.gate_proj.weight", gate[expert_id].contiguous()
                yield f"{base}.{expert_id}.up_proj.weight", up[expert_id].contiguous()
            continue
```

```
        # 处理 down_proj: 按 expert 维度展开
        if name.endswith(".mlp.experts.down_proj.weight") and tensor.dim() == 3:
            base = name.removesuffix(".down_proj.weight")
            for expert_id in range(tensor.size(0)):
                yield f"{base}.{expert_id}.down_proj.weight", tensor[expert_id].contiguous()
            continue

        # 非 MoE 参数或二维张量（密集层）直接透传
        yield name, tensor


class ServerAdapter(BaseRollout):
    ...

    async def update_weights(self, weights, global_steps=None, **kwargs):
        ...
        # 在发送权重前插入 MoE 展开逻辑
        # 跳过 PEFT 已完成基础同步的场景，避免重复展开
        if _should_expand_vllm_moe_params() and not (
            kwargs.get("peft_config") is not None and kwargs.get("base_sync_done", False)
        ):
            weights = _iter_vllm_compatible_moe_params(weights)
        ...
        await sender.async_send_weights(weights)
```

评论区精华

- 关键 Review: gemini-code-assist[bot] 指出原逻辑匹配 endswith(.mlp.experts.gate_up_proj) 漏掉了 PyTorch state dict 中的 .weight 后缀，将导致 MoE 扩展永不触发（critical 级）。作者已在实际提交中修正。
- PR 讨论: 维护者 Luosuu 质疑该问题是否已被新版 vLLM 修复。作者 lxb007981 回应称测试基于 vLLM 0.18.0，而最新修复（PR#47058）尚未进入 0.20.2（verl 0.8.0 指定版本），故该 PR 仍有必要。
- 其他: tardis-key 建议 PR 应作为通用修复而非仅针对 Ascend，已被采纳。
  - suffix 匹配缺失 .weight 后缀 (correctness): 已修改为 endswith(.mlp.experts.gate_up_proj.weight) 等，确保正确匹配。
  - 该修复是否已被新版 vLLM 覆盖 (question): 确认在 verl 0.8.0 指定的 vLLM 0.20.2 中仍需此修复。
  - PR 是否应作为通用修复而非仅针对 Ascend (design): 已被采纳，PR 标题和实现均不特指 Ascend。

风险与影响

- 风险:
  - 版本硬编码边界：_should_expand_vllm_moe_params 固定阈值 0.24.0，若上游 backport 修复到稳定版分支，或用户自行构建修复版本，该条件可能误判，建议补充版

本段下限检查或提供显式配置开关（如 skip_moe_expand）。

- 性能风险：迭代器在 async_send_weights 前展开所有 expert，对大批量 MoE 模型（如 256 experts）可能引入额外 CPU 开销，但因其流式处理且不含额外显存拷贝，影响可控。
- 无测试覆盖：当前无单元测试验证展开逻辑的正确性，也未模拟实际 FSDP-vLLM 同步流程，回归风险依赖手动验证。
- PEFT 路径保护：跳过条件包含 not (peft_config and base_sync_done)，避免重复展开，但若 base_sync_done 语义在后续重构中变化，可能引入微妙问题。

- 影响：
  - 影响范围：限 vllm_rollout.py 一个函数 update_weights，仅在 vLLM <= 0.24.0 且模型为 Qwen-style MoE 时生效；密集模型和已匹配版本无影响。
  - 对用户：解决 Qwen3 MoE + FSDP + vLLM 0.18-0.24 组合下的权重同步崩溃，提升稳定性。
  - 对系统：无额外依赖引入，但需确保 ensure_async_iterator 正确导入。
  - 对团队：为后续 Transformers 5 权重格式变更提供了可复用的扩展模式。
  - 风险标记：版本硬编码边界，缺少测试覆盖，中间件变更

关联脉络

- PR #6906 [rollout] fix: support SGLang FP8 ignored layers for Qwen3.x GatedDeltaNet in rollout: 同为 Qwen3 MoE 模型兼容性修复，但面向 SGLang FP8 忽略层配置，属于同一问题域的不同模块。
- PR #6882 [fully_async] fix: correct the use of partial_rollout: 同为 rollout 模块的 bugfix，涉及权重同步流程。