# PR #6482 完整报告

verl-project/verl

[fsdp, model] feat: support qwen3_5 ulysses sp

合并时间：2026-05-27 11:56

原文链接：http://prhub.com.cn/verl-project/verl/pull/6482

## 执行摘要

- 一句话：支持 Qwen3.5 Ulysses 序列并行
- 推荐动作：建议合并，但有以下后续行动：
  1. 确认 PyTorch 版本：在项目文档或 CI 中明确最低 PyTorch 版本要求，确保 DTensor.full_tensor() 可用。
  2. 考虑性能优化：评估 full_tensor() 的通信开销，若成为瓶颈可考虑改用 to_local() 并在 FSDP 确保下张量已复制的场景下使用。
  3. 补充测试：建议至少添加一个简单的 CPU 测试用例，验证 forward_with_torch_backend 和 forward_with_triton_backend 在启用 Ulysses SP 时的行为。
  4. 精读亮点：该 PR 展示了如何在已有模型架构中为序列并行添加支持，模式可复用于其他模型，值得关注。

## 功能与动机

用户报告使用 Qwen3.5 训练时，启用序列并行（sp=2）会触发 'RuntimeError: The size of tensor a (3058) must match the size of tensor b (1529)'，即序列维度在跨设备切片后未对齐导致的崩溃。该 PR 旨在解决此问题，使 Qwen3.5 模型能够在 FSDP 后端下正常使用 Ulysses 序列并行。关联 issue #5762 明确描述了该问题。

## 实现拆解

1. 修改 verl/models/transformers/qwen3_5.py：
   - 新增导入：从 torch.distributed.tensor 导入 DTensor；从 verl.utils.ulysses 导入 get_ulysses_sequence_parallel_world_size 和 ulysses_pad_and_slice_inputs。
   - 修改 forward_with_torch_backend 函数：在计算 log_probs 和 entropy 之前，先获取 vocab_weights；若其为 DTensor 则调用 .full_tensor() 转为完整张量（DTensor.full_tensor() 是有效方法，用于将分布式张量聚合为完整的本地副本）。然后检查 ulysses_sequence_parallel_size，若大于 1 则对 rolled_labels 调用 ulysses_pad_and_slice_inputs（position_ids_rmpad=None），以在序列维度上进行填充和切片，确保输入对齐。最后将 hidden_states 转换为 vocab_weights 的数据类型。
   - 修改 forward_with_triton_backend 函数：类似地，添加 Ulysses SP 分支：计算 vocab_weights（若为 DTensor 则调用 .full_tensor()），对 rolled_labels 进行

ulysses_pad_and_slice_inputs 切片，并确保 hidden_states 类型一致。

2. 修改 verl/models/transformers/monkey_patch.py：
   - 在 Qwen3.5 猴补丁分支中，导入 Qwen3_5TextModel 和 Qwen3_5MoeTextModel（transformers 库中用于纯文本处理的子模型类）。
   - 当 ulysses_sp_size > 1 时，对这两个文本模型调用 patch_vlm_for_ulysses_input_slicing，以注入序列并行的输入切片逻辑（多模态框架中 VLM 组件的标准处理方式）。
3. 测试与验证：PR body 中附带了实验对比结果截图（sp vs no-sp）并声明已通过验证。

关键文件：

- verl/models/transformers/qwen3_5.py（模块 模型；类别 source；类型 data-contract）：核心实现文件，在 forward_with_torch_backend 和 forward_with_triton_backend 中添加 Ulysses SP 逻辑，处理 DTensor 和序列切片。
- verl/models/transformers/monkey_patch.py（模块 模型；类别 source；类型 data-contract）：猴补丁入口，在 Qwen3.5 分支中为文本模型添加 Ulysses 输入切片补丁。

关键符号：forward_with_torch_backend, forward_with_triton_backend, patch_vlm_for_ulysses_input_slicing

## 关键源码片段

### verl/models/transformers/qwen3_5.py

核心实现文件，在 forward_with_torch_backend 和 forward_with_triton_backend 中添加 Ulysses SP 逻辑，处理 DTensor 和序列切片。

```
# verl/models/transformers/qwen3_5.py
from torch.distributed.tensor import DTensor
from verl.utils.ulysses import (
    get_ulysses_sequence_parallel_world_size,
    ulysses_pad_and_slice_inputs,
)

def forward_with_torch_backend(self, input_ids, labels=None, temperature=1.0, **kwargs):
    outputs = self.model(input_ids, **kwargs)
    hidden_states = outputs[0]
    # 对 labels 进行 roll 操作
    if labels is not None:
        rolled_labels = torch.roll(labels, shifts=-1, dims=-1)
    elif input_ids is not None:
        rolled_labels = torch.roll(input_ids, shifts=-1, dims=-1)
    else:
        raise RuntimeError("...")

    fused_linear_for_ppo = FusedLinearForPPO()
    # 从 lm_head 获取 vocab_weights，如果是 DTensor 则聚合为完整张量
    vocab_weights = self.lm_head.weight
    if isinstance(vocab_weights, DTensor):
        # DTensor.full_tensor() 通过 all-gather 得到完整权重（PyTorch 2.1+）
```

```
        vocab_weights = vocab_weights.full_tensor()

    # 检查 Ulysses SP 是否启用
    ulysses_sequence_parallel_size = get_ulysses_sequence_parallel_world_size()
    if ulysses_sequence_parallel_size > 1:
        # 对 rolled_labels 进行填充和切片，确保序列维度对齐
        rolled_labels, _, _ = ulysses_pad_and_slice_inputs(
            rolled_labels, position_ids_rmpad=None, sp_size=ulysses_sequence_parallel_size
        )
    # 转换 hidden_states 类型以匹配 vocab_weights（通常为 float32）
    hidden_states = hidden_states.to(vocab_weights.dtype)

    log_probs, entropy = fused_linear_for_ppo.forward(
        hidden_states=hidden_states,
        vocab_weights=vocab_weights,
        input_ids=rolled_labels,
        temperature=temperature,
    )
    return Qwen3_5CausalLMOutputForPPO(
        log_probs=log_probs,
        entropy=entropy,
        hidden_states=outputs.hidden_states,
    )
```

## verl/models/transformers/monkey_patch.py

猴补丁入口，在 Qwen3.5 分支中为文本模型添加 Ulysses 输入切片补丁。

```
# verl/models/transformers/monkey_patch.py
# 在 Qwen3.5 猴补丁分支中（以 `elif model.config.model_type in ["qwen3_5", "qwen3_5_moe"]:` 为入口）
from transformers.models.qwen3_5.modeling_qwen3_5 import (
    Qwen3_5ForConditionalGeneration,
    Qwen3_5Model,
    Qwen3_5TextModel, # 新增：纯文本子模型
    Qwen3_5VisionModel,
)
from transformers.models.qwen3_5_moe.modeling_qwen3_5_moe import (
    Qwen3_5MoeForConditionalGeneration,
    Qwen3_5MoeModel,
    Qwen3_5MoeTextModel, # 新增：MoE 纯文本子模型
    Qwen3_5MoeVisionModel,
)

# ... 已有补丁逻辑 ...

# 当启用 Ulysses SP 时，为文本模型注入输入切片逻辑
if ulysses_sp_size > 1:
    patch_vlm_for_ulysses_input_slicing(Qwen3_5TextModel)
    patch_vlm_for_ulysses_input_slicing(Qwen3_5MoeTextModel)
```

## 评论区精华

核心讨论：DTensor.full_tensor() 的正确性

- gemini-code-assist[bot]在 review 中连续两次指出 DTensor 没有 full_tensor() 方法，建议改用 .to_local()，认为这将导致运行时 AttributeError。
- SaltFish11（作者）在第一次评论下回复："DTensor has full_tensor() method."，表明 full_tensor() 是 DTensor 的有效方法（在较新版本 PyTorch 中可用）。
- wuxibin89（合并者）最终 approving review，未进一步质疑，说明接受了作者的辩护或已验证该方法的可用性。
- 风险 / 洞见：该讨论凸显了不同版本 PyTorch API 的差异。full_tensor() 是 PyTorch 2.x 中 DTensor 的合法方法（在 torch>=2.1 中引入），用于将分片张量聚合为完整张量。reviewer 的担心在旧版本中有效，但作者确认当前代码库依赖的版本支持此方法。
  - DTensor.full_tensor() 是否有效 (correctness): 作者确认 full_tensor() 存在且可用，合并者 approving review 说明接受该观点。

## 风险与影响

- 风险：
  1. API 兼容性风险：DTensor.full_tensor() 在较旧 PyTorch 版本（< 2.1）中不存在。如果项目实际使用的 PyTorch 版本低于 2.1，将导致运行时崩溃。但从项目现有代码（其他模型也使用 DTensor）推测，版本要求已满足。
  2. 性能开销：full_tensor() 会触发 all-gather 通信，在每步 forward 中复制 lm_head.weight（即 vocab 投影矩阵）到所有设备上，带来额外通信开销。对于超大词表（如 256k）模型，这可能成为瓶颈。
  3. 回归风险：当 ulysses_sequence_parallel_size = 1（未启用 SP）时，新增代码不执行任何额外操作，逻辑与旧版一致，回归风险低。
  4. 多模态兼容性：patch_vlm_for_ulysses_input_slicing 应用于文本模型组件，但未测试输入包含图像时的行为。若文本模型与视觉模型共享嵌入层或其他组件，可能出现未预期交互。
  5. 测试覆盖不足：本次变更未提供对应的单元测试或集成测试，仅依赖作者的手动实验验证。 - 影响：影响范围：限于 Qwen3.5 模型（含 MoE 变体）在 FSDP 后端下的训练流程。影响程度：中。修复了使用序列并行时的崩溃问题，使该功能正常工作。对不使用 Qwen3.5 或不启用 SP 的用户无影响。性能影响：启用 SP 后，由于通信开销，可能微有性能下降，但这是序列并行本身的特性，非本 PR 引入。
- 风险标记：API 兼容性风险，性能开销，测试覆盖不足

## 关联脉络

- PR #5682 Search for similar PRs: PR body 中引用了该 PR 作为类似变更的搜索参考，可能实现了其他模型的 Ulysses SP 支持。
- PR #6488 [veomni] fix: VeOmniEngineWithValueHead loads ForTokenClassification: 同属模型修复和特性支持，涉及引擎适配。