PR #6660 完整报告

verl-project/verl

[model, fsdp] fix: support Qwen3.5 linear attention under Ulysses SP

合并时间：2026-07-20 14:44

原文链接：http://prhub.com.cn/verl-project/verl/pull/6660

执行摘要

- 一句话：修复 Qwen3.5 线性注意力 Ulysses SP 序列边界问题
- 推荐动作：值得精读。该 PR 展示了如何在 FSDP 框架下支持自定义模型的 packed 训练，涉及 monkey patch、分布式上下文构建（CP context）、签名检测保持模型无关性等技巧。特别关注 _build_fla_cp_context 和 _prepend_cp_conv_prefix 的设计，以及如何在不破坏引擎通用性的前提下传递 packed 序列边界。

功能与动机

Current Qwen3.5/Qwen3.5-MoE training with FSDP remove-padding + Ulysses SP does not pass packed sequence boundaries into linear attention, so packed samples can share linear-attention state across sequence boundaries; the Qwen3.5 fused PPO loss can also re-slice already SP-sliced labels.

实现拆解

1. 模型层 packed 支持（verl/models/transformers/qwen3_5.py）：新增辅助函数 _prepare_packed_seq_idx、_split_packed_args、_build_fla_cp_context 等，用于生成 packed 序列索引、在 Ulysses SP 下构建 CP context。重写 qwen3_5_decoder_layer_forward 和 qwen3_5_gated_delta_net_forward，使其在 SP 时使用 FLA 的 packed/CP 内核，在 conv1d fallback 路径中通过 _split_packed_args 按序列维度拆分 packed 张量。同时修复 forward_with_torch_backend 中重复 slicing shift_labels 的问题。
2. Monkey patch 注册（verl/models/transformers/monkey_patch.py）：将新的 forward 方法安装到 Qwen3_5DecoderLayer、Qwen3_5MoeDecoderLayer、Qwen3_5GatedDeltaNet、Qwen3_5MoeGatedDeltaNet，确保运行时调用 packed 感知版本。
3. FSDP 引擎改动（verl/workers/engine/fsdp/transformer_impl.py）：在 _build_model_optimizer 中通过 signature(module.forward) 检测是否接受 cu_seqlens 参数，动态设置 pass_packed_cu_seqlens。在 prepare_model_inputs 中，当 pad_mode==NO_PADDING 时，从 TensorDict offset 获取 packed_cu_seqlens，在 Ulysses SP 激活时补全填充长度后传入 model inputs。此方式保持引擎模型无关。
4. 单元测试补充（tests/models/test_fsdp_no_padding_on_gpu.py）：新增 test_prepare_model_inputs_passes_declared_packed_sequence_boundaries_on_gpu，

验证 engine 正确传递 cu_seqlens。

5. 分布式回归测试（tests/special_distributed/test_qwen35_linear_attention_ulysses_sp.py）：新增完整测试，覆盖边界对齐（boundary_aligned）、序列切分（sequence_cut）、单序列（single_sequence）、多短序列（many_short_sequences）场景，同时验证 causal_conv1d_fn 和 torch.conv1d 回退路径的输出与梯度误差。

关键文件：

- verl/models/transformers/qwen3_5.py（模块 模型层；类别 source；类型 core-logic；符号 _call_accepts_kwarg, _prepare_packed_seq_idx, _split_packed_args, _ConvPrefixExchange）：核心模型层变更，新增 packed sequence 边界处理、CP context 构建、conv prefix 交换等逻辑，重写 DecoderLayer 和 GatedDeltaNet forward 方法。
- tests/special_distributed/test_qwen35_linear_attention_ulysses_sp.py（模块 测试；类别 test；类型 test-coverage；符号 _err_ratio, _make_layer, _broadcast_params, _all_gather_seq）：新增分布式回归测试，覆盖多种 packed 场景，确保修复正确性。
- verl/workers/engine/fsdp/transformer_impl.py（模块 引擎；类别 source；类型 dependency-wiring）：FSDP 引擎修改，动态检测模型 forward 是否接受 cu_seqlens，并在 prepare_model_inputs 中传递 packed 边界，同时保持模型无关性。
- tests/models/test_fsdp_no_padding_on_gpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_prepare_model_inputs_passes_declared_packed_sequence_boundaries_on_gpu）：GPU 单元测试验证 FSDP 引擎能正确传递 packed 序列边界。
- verl/models/transformers/monkey_patch.py（模块 补丁；类别 source；类型 configuration）：安装 Qwen3.5 decoder 和 gated delta net 的 monkey patch。

关键符号：_prepare_packed_seq_idx, _split_packed_args, _build_fla_cp_context, _prepend_cp_conv_prefix, qwen3_5_decoder_layer_forward, qwen3_5_gated_delta_net_forward

关键源码片段

verl/models/transformers/qwen3_5.py

核心模型层变更，新增 packed sequence 边界处理、CP context 构建、conv prefix 交换等逻辑，重写 DecoderLayer 和 GatedDeltaNet forward 方法。

```
# 关键工具函数：按 cu_seqlens 拆分 packed 张量
# tensors：需要拆分的张量元组，dim 指定序列维度
# cu_seqlens_cpu 避免重复 GPU-CPU 同步
def _split_packed_args(
    cu_seqlens: torch.LongTensor,
    tensors: tuple[torch.Tensor, ...],
    cu_seqlens_cpu: Optional[torch.LongTensor] = None,
    dim: int = 1,
):
    offsets = (cu_seqlens_cpu if cu_seqlens_cpu is not None else cu_seqlens.detach().cpu()).
    tolist()
```

```
    for start, end in zip(offsets[:-1], offsets[1:], strict=True):
        chunks = []
        for tensor in tensors:
            split_dim = dim if dim >= 0 else tensor.ndim + dim
            slices = [slice(None)] * tensor.ndim
            slices[split_dim] = slice(start, end)
            chunks.append(tensor[tuple(slices)])
        yield tuple(chunks)


# 构建 FLA 的 Context Parallel (CP) context
# 在 Ulysses SP 下用于在线性注意力中传递 conv 前缀
def _build_fla_cp_context(
    cu_seqlens: torch.LongTensor,
    cu_seqlens_cpu: Optional[torch.LongTensor],
    conv_kernel_size: int,
):
    group = get_ulysses_sequence_parallel_group()
    if group is None:
        return None
    from fla.ops.cp.context import build_cp_context
    return build_cp_context(
        cu_seqlens=cu_seqlens,
        cu_seqlens_cpu=cu_seqlens_cpu,
        group=group,
        conv1d_kernel_size=conv_kernel_size,
    )


# qwen3_5_decoder_layer_forward 的核心 packed 路径（简化）
if hidden_states is not None and model_cu_seqlens is not None:
    # 构建 CP context 用于跨 rank 的 conv prefix 交换
    cp_context = _build_fla_cp_context(model_cu_seqlens, model_cu_seqlens_cpu, conv_kernel_
    size)
    if cp_context is not None:
        # 在 mixed_qkv 前附加从上一 rank 接收的 conv prefix
        mixed_qkv, conv_prefix_len = _prepend_cp_conv_prefix(mixed_qkv, cp_context)
        # 调整 cu_seqlens 以反映 prefix 长度
        model_cu_seqlens = model_cu_seqlens + conv_prefix_len
        model_cu_seqlens[0] = 0
        model_cu_seqlens_cpu = model_cu_seqlens_cpu + conv_prefix_len if model_cu_seqlens_
        cpu is not None else None
        model_cu_seqlens_cpu[0] = 0 if model_cu_seqlens_cpu is not None else 0

    # 使用 FLA packed 内核处理线性注意力
    if _call_accepts_kwarg(self.self_attn.chunk_gated_delta_rule, "cp_context"):
        attn_output, _ = self.self_attn.chunk_gated_delta_rule(
            query, key, value,
            cp_context=cp_context,
```

```
            cu_seqlens=model_cu_seqlens,
            ...
        )
    else:
        # fallback: 按 packed 序列逐个处理
        for (q_i, k_i, v_i) in _split_packed_args(
            model_cu_seqlens, (query, key, value),
            cu_seqlens_cpu=model_cu_seqlens_cpu, dim=2 # 序列维度
        ):
            out_i, _ = self.self_attn.chunk_gated_delta_rule(q_i, k_i, v_i, ...)
            outputs.append(out_i)
```

verl/workers/engine/fsdp/transformer_impl.py

FSDP 引擎修改，动态检测模型 forward 是否接受 cu_seqlens，并在 prepare_model_inputs 中传递 packed 边界，同时保持模型无关性。

```
# _build_model_optimizer 中动态检测 model forward 签名
# 当 forward 声明接受 cu_seqlens 参数时，后续准备输入时传入 packed 边界
def _build_model_optimizer(self):
    module = self._build_module()
    # 通过反射检查 model forward 是否接受 cu_seqlens 参数
    try:
        self.pass_packed_cu_seqlens = "cu_seqlens" in signature(module.forward).parameters
    except (TypeError, ValueError):
        self.pass_packed_cu_seqlens = False
    # ... 后续代码保持原有


# prepare_model_inputs 中注入 cu_seqlens 和 cu_seqlens_cpu
def prepare_model_inputs(self, micro_batch: TensorDict):
    # ... 前面获取 input_ids, position_ids 等
    pass_packed_cu_seqlens = getattr(self, "pass_packed_cu_seqlens", False)

    packed_cu_seqlens = None
    if pad_mode == DatasetPadMode.NO_PADDING:
        input_ids_rmpad = input_ids.values().unsqueeze(0) # (1, total_nnz)
        # 从 TensorDict 的 offsets() 获取 packed sequence 边界
        packed_cu_seqlens = input_ids.offsets().to(
            device=input_ids_rmpad.device, dtype=torch.long
        )
        # ... 其他 remove-padding 处理

    # 当启用 Ulysses SP 时记录填充大小，用于扩展 cu_seqlens
    sp_pad_size = 0
    if self.use_ulysses_sp:
        # ... 执行 pad and slice
        sp_pad_size = pad_size

    # 如果模型 forward 声明了 cu_seqlens，则将其加入 model_inputs
```

```
if packed_cu_seqlens is not None and pass_packed_cu_seqlens:
    model_cu_seqlens = packed_cu_seqlens
    if self.use_ulysses_sp and sp_pad_size:
        # SP 填充后总长度增加，追加一个包含填充总长的边界
        padded_total = int(model_cu_seqlens[-1].item()) + int(sp_pad_size)
        model_cu_seqlens = torch.cat([
            model_cu_seqlens,
            model_cu_seqlens.new_tensor([padded_total]),
        ])
    model_inputs["cu_seqlens"] = model_cu_seqlens
    model_inputs["cu_seqlens_cpu"] = model_cu_seqlens.cpu()
```

评论区精华

- 维度切片错误：gemini-code-assist 指出 _packed_causal_conv1d_fallback 中 slicing 沿 hidden 维度而非 sequence 维度，且建议传入 cu_seqlens_cpu 避免重复 GPU-to-CPU 拷贝。作者在 commit bac87e7f 中修复，_split_packed_args 新增 dim 和 cu_seqlens_cpu 参数，conv 回退路径传入 dim=2。
- 过度防御 try-except：bot 建议移除 _call_accepts_kwarg 中包裹 signature 的 try-except，因为纯 Python 函数不会抛出异常。作者同样在 bac87e7f 中移除。
- 引擎模型特定检查：Luosuu 和 wuxibin89 指出 transformer_impl.py 不应硬编码 Qwen3.5 模型类型。作者改为通过反射检测 forward 签名是否接受 cu_seqlens，在 commit 3f1ecb21 中实现，保持引擎模型无关。
  - _split_packed_args 在 conv fallback 中维度错误 (correctness): 作者在 commit bac87e7f 中修复，_split_packed_args 现接受 dim 和 cu_seqlens_cpu 参数，conv fallback 传递 dim=2。
  - _call_accepts_kwarg 中的过度防御 (style): 作者移除防御性捕获，使用直接签名检查。
  - transformer_impl.py 应保持模型无关 (design): 作者改为通过反射检测 forward 签名中是否包含 cu_seqlens，动态决定传递 packed 序列，保持模型无关。

风险与影响

- 风险：
  - 回归风险：monkey patch 替换了 Qwen3.5 标准 forward，若新实现与 HuggingFace 预期不一致可能导致输出错误。但分布式测试覆盖了多种 packed 场景，误差比（3.6e-4～1.3e-3）在可接受范围。
  - 性能风险：每层调用 _split_packed_args 拆分层带来额外开销，但 FLA packed kernels 通常更高效；回退路径经过验证。
  - 兼容性风险：依赖 FLA 库（≥0.5.1），若 FLA 不可用则使用纯 PyTorch fallback，均已测试。仅影响 Qwen3.5/Qwen3.5-MoE 模型类型。
  - 签名检测稳定性：signature(module.forward) 对动态构造的 callable 可能失败，代码已捕获 TypeError/ValueError。
- 影响：

- 用户影响：使用 Qwen3.5 的用户现在可以安全启用 use_remove_padding=True + ulysses_sequence_parallel_size>1 进行训练，无需担心线性注意力状态泄露。
- 系统影响：FSDP 引擎增加少量反射检测逻辑，对其他模型无影响并保持模型无关。
- 团队影响：为后续模型添加类似 packed 支持提供了可复用的模式（签名检测 + monkey patch）。
- 风险标记：核心路径变更，依赖 FLA 库，monkey patch 风险，兼容性风险

关联脉络

- 暂无明显关联 PR