

PR #7221 完整报告

verl-project/verl

[megatron] feat: support contiguous context-parallel layout for DeepSeek V4

合并时间: 2026-08-01 12:23

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7221>

执行摘要

- 一句话: Megatron 新增 contiguous CP 布局支持 DeepSeek V4
- 推荐动作: 值得精读。这是 Megatron THD 数据通路的核心切片逻辑变更, 展示了如何为特定注意力变体扩展序列并行布局而不破坏默认路径。关注点: `_get_context_parallel_layout` 的集中决策方式、contiguous 切片的区间求交写法、以及 `postprocess` 的按序拼接反解。对需要在 Megatron 后端接入新模型并行语义的工程师有直接参考价值。上线前需要补齐 DS4 端到端回归与更大 CP 规模验证。

功能与动机

PR body 明确指出 DeepSeek V4 的注意力期望每个 CP rank 拥有全局 padded THD buffer 的一个连续区间, 而现有 zigzag 布局给每个 rank 的是从序列两端取出的块, "Running DS4 under the zigzag layout silently produces wrong attention results." 即静默产生错误注意力结果, 因此需要引入 contiguous 布局并自动为 DS4 启用, 同时保持默认 zigzag 以向后兼容。

实现拆解

1. THD 预处理入口扩展 (verl/models/mcore/util.py) : `preprocess_thd_engine` 与 `postprocess_thd_engine` 新增 `cp_layout: Literal["zigzag", "contiguous"] = "zigzag"` 参数, 并在入口校验非法值直接抛 `ValueError`。对齐因子改为 `tp_size * cp_size * cp_layout_factor`, 其中 `cp_layout_factor` 仅在 zigzag 且 `cp_size > 1` 时为 2, 避免 contiguous 布局下的过度 padding。
2. contiguous 切片与独立 roll: 在 `preprocess_thd_engine` 中, 每个 rank 计算自己在全局 padded buffer 上的区间 `[shape[0] * cp_rank, shape[0] * (cp_rank + 1))`, 逐序列求交集后拷贝; 跳过与本地区间无交集的序列。需要 roll 时, 用 `(idx + 1) % seqlen` 逐序列独立 roll, 不再做 zigzag 所需的全局 `torch.roll`。
3. postprocess 反解: `postprocess_thd_engine` 在 contiguous 模式下按 CP rank 顺序 `torch.cat` 各 rank 输出, 再按 `cu_seqLens` 切回每条有效序列, 恢复原始嵌套张量结构; zigzag 路径保持不变。
4. 引擎层统一布局决策 (verl/workers/engine/megatron/transformer_impl.py) : `MegatronEngine` 新增静态方法 `_get_context_parallel_layout(unwrapped_model)`, 按 `experimental_attention_variant == "dsv4_hybrid"` 返回 contiguous, 否则返回 zigzag。LM head 的 `forward_step` (fused 与非 fused) 和 value head 的 `forward_step` 都调用该方法并把 `cp_layout` 传给前向函数, 避免两条路径漂移。

5. 前向路径透传 (verl/models/mcore/model_forward.py、model_forward_fused.py) :
gptmodel_forward_model_engine 与 fused_forward_model_engine_inner 增加
cp_layout 参数, 并在所有 preprocess_thd_engine / postprocess_thd_engine 调用点透
传。测试配套: 新增 / 扩展 CPU 单测覆盖区间选择、独立 roll、postprocess 反解和非法
布局报错, value head 布局透传测试保证引擎层改动被覆盖。

关键文件:

- verl/models/mcore/util.py (模块 序列预处理; 类别 source; 类型 core-logic; 符号
preprocess_thd_engine, postprocess_thd_engine, ContextParallelLayout) : THD 序列
预处理 / 后处理的核心实现, 新增 cp_layout 参数以支持 contiguous 布局, 是本次变更的
技术核心。
- verl/workers/engine/megatron/transformer_impl.py (模块 引擎层; 类别 source; 类型
core-logic; 符号 _get_context_parallel_layout) : MegatronEngine 新增
_get_context_parallel_layout(), 统一 policy 与 value head 的 CP 布局选择, 避免两条路
径漂移。
- tests/utils/test_megatron_bshd_preprocess.py (模块 测试; 类别 test; 类型
test-coverage; 符号 _load_mcore_util_with_stubbed_megatron,
test_preprocess_thd_engine_contiguous_uses_global_packed_intervals,
test_preprocess_thd_engine_rejects_unknown_cp_layout,
test_preprocess_thd_engine_contiguous_rolls_each_sequence_independently) : 覆盖
contiguous 预处理的区间选择、逐序列 roll、postprocess 反解与非法布局拒绝。
- tests/workers/test_megatron_value_head_cp_layout_on_cpu.py (模块 测试; 类别 test
; 类型 test-coverage; 符号 test_dsv4_value_head_forwards_contiguous_cp_layout) :
验证 value head 前向会把 cp_layout='contiguous' 透传给 forward_fn, 防止
policy/value 路径不一致。
- verl/models/mcore/model_forward.py (模块 前向路径; 类别 source; 类型
data-contract; 符号 gptmodel_forward_model_engine) : 非 fused 前向路径透传
cp_layout 到预处理 / 后处理。
- verl/models/mcore/model_forward_fused.py (模块 前向路径; 类别 source; 类型
data-contract; 符号 fused_forward_model_engine_inner) : fused kernel 前向路径同样
需要透传 cp_layout。

关键符号: preprocess_thd_engine, postprocess_thd_engine,
_get_context_parallel_layout, gptmodel_forward_model_engine,
fused_forward_model_engine_inner

关键源码片段

verl/models/mcore/util.py

THD 序列预处理 / 后处理的核心实现, 新增 cp_layout 参数以支持 contiguous 布局, 是本次
变更的技术核心。

```
# verl/models/mcore/util.py — contiguous 布局下每个 rank 取全局 padded buffer 的连续区间  
local_global_start = shape[0] * cp_rank
```

```

local_global_end = local_global_start + shape[0]

for i in range(batch_size):
    if cp_layout == 'contiguous':
        # 本条序列在全局 buffer 中的起点与有效终点 (不含 padding)
        seq_global_start = cu_seqlens_padded_cpu[i]
        seq_valid_end = seq_global_start + seqlens_in_batch_cpu[i]
        # 与本 rank 区间求交; 完全不相交则跳过
        copy_global_start = max(local_global_start, seq_global_start)
        copy_global_end = min(local_global_end, seq_valid_end)
        if copy_global_start >= copy_global_end:
            continue

        # 换算为序列内偏移与本地缓冲内偏移
        source_start = copy_global_start - seq_global_start
        source_end = copy_global_end - seq_global_start
        destination_start = copy_global_start - local_global_start
        destination_end = copy_global_end - local_global_start
        d = input_ids[i]

        if need_roll:
            # contiguous 布局按序列独立 roll: (idx + 1) % seqlen
            source_indices = torch.arange(source_start, source_end, device=d.device)
            source_indices = (source_indices + 1) % seqlens_in_batch_cpu[i]
            input_ids_rmpad[destination_start:destination_end] = torch.index_select(d, 0, source_indices)
            position_ids_rmpad[destination_start:destination_end] = source_indices
        else:
            input_ids_rmpad[destination_start:destination_end] = d[source_start:source_end]
            position_ids_rmpad[destination_start:destination_end] = torch.arange(
                source_start, source_end, dtype=torch.long, device=input_ids.device
            )
        continue

```

verl/workers/engine/megatron/transformer_impl.py

MegatronEngine 新增 `_get_context_parallel_layout()`, 统一 policy 与 value head 的 CP 布局选择, 避免两条路径漂移。

```

# verl/workers/engine/megatron/transformer_impl.py — 统一 CP 布局决策
@staticmethod
def _get_context_parallel_layout(unwrapped_model) -> str:
    # DeepSeek V4 的注意力通过 rank 本地 global_start 寻址行,
    # 必须使用 contiguous 布局; 其余模型保持 zigzag 以利用因果掩码负载均衡。
    if getattr(unwrapped_model.config, 'experimental_attention_variant', None) == 'dsv4_hybrid':
        return 'contiguous'
    return 'zigzag'

# forward_step 中, LM head 与 value head 统一从这里取布局再透传
unwrapped_model = unwrap_model(model)

```

```
cp_layout = self._get_context_parallel_layout(unwrapped_model)
```

评论区精华

该 PR 未收到人工 review 评论，仅有 Gemini 机器人提示服务停用的自动化评论。设计权衡集中在 PR body:

1) roll 语义按布局拆分，全局 `torch.roll` 仅用于 zigzag, contiguous 逐序列独立 roll; 2) 未知 `cp_layout` 显式抛 `ValueError` 而非静默回退; 3) 双 GPU 验证以 CP1 单卡为参考，并额外运行两次 CP1 测量确定性下限。

- 暂无高价值评论线程

风险与影响

- 风险：端到端验证缺口：PR body 明确说明没有覆盖经过 DeepSeek V4 model provider 的 SFT 路径，只验证了原生 mcore 模块；verl main 不存在不依赖 Megatron-Bridge 的 deepseek_v4 provider。CP 规模覆盖有限：仅验证 CP 1 和 CP 2，更大 CP 规模未实测，跨序列 straddle 行为在更大 CP 下可能有边界问题。负载均衡特性变化：zigzag 专为因果掩码负载均衡设计（每个 rank 拿序列两端），contiguous 会使长序列的尾部区间集中到部分 rank，DS4 下这是内核要求，但长尾场景的显存 / 计算倾斜需要观察。动态 CP 组合未验证：local_cp_size（动态 CP）路径与 cp_layout 的组合没有测试，如果 DS4 与动态 CP 一起使用需要额外验证。核心数据通路变更：preprocess_thd_engine / postprocess_thd_engine 是 Megatron THD 训练的统统一入口，虽然默认分支未动，但任何对公共函数的修改都值得在常规模型上跑一次回归。
- 影响：影响范围：主要影响 Megatron 后端 + THD (use_remove_padding) 路径，对使用 experimental_attention_variant == 'dsv4_hybrid' 的 DeepSeek V4 用户，CP 布局自动变为 contiguous；其他模型与 cp_size == 1 场景完全不受影响。同时改动多个模型前向入口的数据契约（新增可选参数），对内部 API 调用方有传导。团队影响：为 DS4 在 verl 的训练支持补齐关键的一块拼图。
- 风险标记：核心预处理路径变更，端到端 DS4 训练未验证，CP 规模大于 2 未覆盖，contiguous 改变负载均衡特性，动态 CP 与 cp_layout 组合未验证

关联脉络

- PR #7101 [docker] feat: upgrade vllm and megatron version, add packages to support DeepSeek-V4: 同一 DeepSeek V4 支持功能线，该 PR 升级 vLLM 与 Megatron 依赖版本，为 DS4 运行提供环境基础。
- PR #7181 [megatron] feat: delta_sharded on Megatron-Bridge param mappings (TP+EP, hybrid-Mamba): 同样修改 verl/workers/engine/megatron/transformer_impl.py，同属 Megatron 后端能力建设。
- PR #7144 [ckpt, fsdp] feat: sharded delta block placements + backend-owned HF export (engine core + FSDP): 同样改动 engine core (transformer_impl.py)，与模型并行布局和引擎层演进同线。