PR #7407 完整报告

verl-project/verl

[megatron,veomni] feat: use torch.int16 for routed_experts

合并时间：2026-08-14 18:39

原文链接：http://prhub.com.cn/verl-project/verl/pull/7407

执行摘要

- 一句话：路由索引改 int16，支持超 255 专家并重构回放
- 推荐动作：值得精读。亮点包括：NCCL 无 int16 时的 uint8 位视图通信模式、用 duplicate-top-k 检测替代显式 replay_mask 的设计取舍、controller 与 engine 的职责划分（controller 不感知形状）。建议阅读时对照 tests/utils/test_megatron_router_replay_dcp.py 理解通信层约束，并对照 verl/workers/utils/padding.py 的删除逻辑确认数据契约变更范围。

功能与动机

PR body 的目标是 Use torch.int16 for routed_experts to reduce memory footprint。结合代码证据，真实动机有两层：一是正确性——uint8 只能表达 0-255 的 expert id，DeepSeek-V4 等新一代 MoE 的专家规模可能超过 255，旧代码在 padding.py 中条件压缩、在 router replay 记录中固定 to(uint8)，会产生静默截断（新增测试 test_record_output_preserves_ids_above_uint8_range 专门回归 256/511/1023 等 id）；二是内存——int16 相比 int64 把路由索引的显存占用降到 1/4。移除 replay_mask 的动机在模块 docstring 中有说明：所有"无记录路由"的位置（pad 后缀、R3 rollout 未记录的零行）都以全零目标行到达，控制器用 duplicate-top-k 检测即可原生回退，不再需要单独 mask 传递。

实现拆解

1. 数据契约提升为 int16：verl/workers/utils/padding.py 的 left_right_2_no_padding 删除 routed_experts.max() <= 255 时转 uint8 的条件分支，避免记录阶段截断；verl/utils/megatron/router_replay_utils.py 的 merge_router_topk_indices 记录时改为 to(torch.int16)，SP all-gather 前用 contiguous().view(torch.uint8) 绕行 NCCL 无 int16 的限制，postprocess 后再 view(torch.int16) 还原（嵌套路径用 nested_tensor_from_jagged(values.view(int16), offsets)）；merge_nested_router_maps 与 pp_gather 同步调整，pp_gather 非嵌套分支用 uint8 view 做 all_gather 后还原；verl/models/mcore/util.py 的 postprocess_thd_engine 让中间 torch.empty 继承 output.dtype；verl/workers/engine/megatron/transformer_impl.py 移除 pp_gather 前的 .to(torch.uint8)。
2. VeOmni router replay 控制器重构：verl/utils/veomni/router_replay.py 由多 micro-batch + 显式 replay_mask 简化为单 micro-batch 语义——_recorded 从 list[list[Tensor]] 变为 list[Tensor]；begin_microbatch(targets=...) 取代

set_microbatch_targets 与 advance_record_microbatch；take_recorded() 取代 collect_recorded(...)；新增 num_fired/num_targets 属性供引擎在 REPLAY 后做契约断言；移除内部 sp_group 依赖与 all-gather/pad/slice 逻辑。

3. VeOmni 引擎装配调整：verl/workers/engine/veomni/transformer_impl.py 删除 forward_backward_batch 内 side-channel 聚合逻辑，新增 _prepare_router_replay_inputs（REPLAY 时把 routed_experts 按 input_ids 同一 pad + Ulysses 规则切分并按层 unbind），prepare_model_outputs 在 RECORD 时执行 take_recorded().view(uint8) -> _gather_and_unpad_packed -> view(int16) -> nested_tensor_from_jagged，并新增 REPLAY 后 num_fired != num_targets 的严格报错。
4. 配套修复：verl/utils/reward_score/math_dapo.py 的 verify 在 Minerva 提取失败（pred == "[INVALID]"）时回退到 \boxed{} 严格提取；verl/utils/tokenizer/deepseek.py 与 verl/experimental/agent_loop/agent_loop.py 修复 DeepSeekV4ContinuousTokenBuilder 相关逻辑；examples/grpo_trainer/run_deepseek_v4_veomni.sh 修正脚本。
5. 测试配套：tests/workers/test_router_replay_engine_helpers_on_cpu.py 重写为覆盖 _prepare_router_replay_inputs 与 prepare_model_outputs（含 >255 id 保真）；tests/utils/veomni/test_router_replay_on_cpu.py 迁移到新 API（begin_microbatch/take_recorded）；tests/utils/test_megatron_router_replay_dcp.py 验证 uint8 view 通信与终态 int16；agent_loop schema 测试同步更新。

关键文件：

- verl/utils/veomni/router_replay.py（模块 路由回放；类别 source；类型 core-logic；符号 init, num_fired, num_targets, begin_microbatch）：VeOmni router replay 控制器核心重构：从多 micro-batch + replay_mask 简化为单 micro-batch + duplicate-top-k 回退，是 PR 移除 mask 方案与 int16 落地的关键。
- verl/workers/engine/veomni/transformer_impl.py（模块 引擎层；类别 source；类型 core-logic；符号 _prepare_router_replay_inputs, prepare_model_outputs, forward_backward_batch）：引擎侧装配重构：删除 side-channel 聚合，新增 _prepare_router_replay_inputs 与 RECORD 路径的 gather/unpad/nested 重包，并加入 REPLAY 层数契约校验。
- verl/utils/megatron/router_replay_utils.py（模块 路由回放；类别 source；类型 core-logic；符号 merge_router_topk_indices, merge_nested_router_maps, pp_gather）：Megatron 侧 int16 数据契约落地点：SP/PP 通信通过 uint8 位视图绕行 NCCL 无 int16 的限制，是跨并行通信的关键改动。
- verl/workers/utils/padding.py（模块 数据处理；类别 source；类型 data-contract；符号 left_right_2_no_padding）：删除 routed_experts 的 uint8 条件压缩分支，是修复 expert id 超过 255 被截断的直接入口，影响所有后续 dtype 契约。
- verl/models/mcore/util.py（模块 模型工具；类别 source；类型 data-contract；符号 postprocess_thd_engine）：postprocess_thd_engine 中间张量透传 dtype，保证 uint8 payload 在 CP 拼接时不被强制为 float32，是 Megatron 侧 int16 方案成立的前提。
- verl/workers/engine/megatron/transformer_impl.py（模块 引擎层；类别 source；类型 data-contract；符号 forward_backward_batch）：移除 pp_gather 前的 to(uint8) 强制转

换，与上游 int16 数据契约对齐。

- verl/utils/reward_score/math_dapo.py（模块 奖励函数；类别 source；类型 bugfix；符号 verify, compute_score）：DAPO 奖励函数增加 Minerva 失败后回退到 \boxed{} 严格校验，提升 reward 提取鲁棒性。
- tests/workers/test_router_replay_engine_helpers_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_jagged_routed_experts, TestPrepareRouterReplayInputs, TestPrepareModelOutputs）：重写为覆盖新引擎 helper（_prepare_router_replay_inputs / prepare_model_outputs），其中 >255 expert id 保真测试直接验证本 PR 的核心修复。
- tests/utils/veomni/test_router_replay_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_take_recorded_is_layer_major, test_record_begin_microbatch_drops_previous_microbatch, test_replay_strict_missing_target_pos_raises）：控制器状态机测试迁移到新 API（begin_microbatch/take_recorded），并移除 replay_mask 相关用例。
- tests/utils/test_megatron_router_replay_dcp.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_router_record_and_replay_use_dynamic_cp_size, test_pp_gather_normalizes_nested_routes_to_cpu）：验证 Megatron 侧 uint8 view 通信路径与终态 int16 dtype，是对通信绕行方案的关键回归测试。

关键符号：VeOmniRouterReplay.begin_microbatch, VeOmniRouterReplay.take_recorded, VeOmniRouterReplay.on_router_forward, VeOmniRouterReplay.num_fired, VeOmniRouterReplay.num_targets, VeOmniEngineWithLMHead._prepare_router_replay_inputs, VeOmniEngineWithLMHead.prepare_model_outputs, merge_router_topk_indices, merge_nested_router_maps, pp_gather, left_right_2_no_padding, postprocess_thd_engine, verify

关键源码片段

verl/utils/veomni/router_replay.py

VeOmni router replay 控制器核心重构：从多 micro-batch + replay_mask 简化为单 micro-batch + duplicate-top-k 回退，是 PR 移除 mask 方案与 int16 落地的关键。

```
# verl/utils/veomni/router_replay.py
# 重构后控制器只持有单个 micro-batch 的状态，形状逻辑全部下沉到引擎。
def __init__(self) -> None:
    self._action: RouterReplayAction = RouterReplayAction.DISABLED
    # id(router_module) -> position，每个 micro-batch 重建；模块不在表中
    # 即代表“首次 fire”，backward recompute 复用同一 id 命中同一位置。
    self._id_to_pos: dict[int, int] = {}
    # RECORD：每层位置一个 [nnz, topk] 张量，按 fire 顺序存放。
    self._recorded: list[torch.Tensor] = []
    # REPLAY：每层位置一个 [nnz, topk] 目标张量。
    self._targets: list[torch.Tensor] = []
    # 环境变量门控的形状调试检查。
    self._debug: bool = os.environ.get("VERL_ROUTER_REPLAY_DEBUG") == "1"
```

```
        self._installed: bool = False

    @property
    def num_fired(self) -> int:
        """自 begin_microbatch 以来实际 fire 过的 router 数量，用于引擎侧契约断言。"""
        return len(self._id_to_pos)

    @property
    def num_targets(self) -> int:
        """本次 micro-batch 的 REPLAY 层目标数量。"""
        return len(self._targets)
```

verl/workers/engine/veomni/transformer_impl.py

引擎侧装配重构：删除 side-channel 聚合，新增 _prepare_router_replay_inputs 与 RECORD 路径的 gather/unpad/nested 重包，并加入 REPLAY 层数契约校验。

```
    # verl/workers/engine/veomni/transformer_impl.py
    # REPLAY 目标装载：与 input_ids 同规则 pad + Ulysses slice，再按层拆给控制器。
    def _prepare_router_replay_inputs(self, micro_batch: TensorDict, output_args: dict) -> None:
        rr = self._router_replay
        if rr is None or rr.action is RouterReplayAction.DISABLED:
            return

        if rr.action is RouterReplayAction.RECORD:
            rr.begin_microbatch()
            return

        routed = micro_batch.get("routed_experts", None)
        if routed is None:
            raise RuntimeError(
                "router_replay REPLAY: micro_batch missing 'routed_experts'. "
                "Verify that compute_log_prob (R2) or the rollout path (R3) "
                "attached routed_experts to the batch before this engine "
                "call, and that left_right_2_no_padding preserved it."
            )

        # Nested-jagged [bs, seq, L, topk] -> rmpad [total_nnz, L, topk]。
        # 统一抬到 int64，避免 pad 拼接与索引操作受 dtype 截断影响。
        targets = (routed.values() if routed.is_nested else routed).to(torch.int64)
        pad_size = int(output_args.get("pad_size", 0))
        if pad_size:
            # 补上与 input_ids 相同的 pad 后缀；全零行在控制器内触发
            # duplicate-top-k 回退，等价于旧实现里显式 mask 的作用。
            targets = torch.cat([targets, targets.new_zeros((pad_size, *targets.shape[1:]))])
        if self.use_ulysses_sp:
            targets = slice_input_tensor(targets, dim=0, padding=False)

        # 按层维度拆成 L 个 [total_nnz, topk] 目标，控制器按 router id 延迟匹配位置。
        rr.begin_microbatch(targets=list(targets.unbind(dim=1)))
```

verl/utils/megatron/router_replay_utils.py

Megatron 侧 int16 数据契约落地点：SP/PP 通信通过 uint8 位视图绕行 NCCL 无 int16 的限制，是跨并行通信的关键改动。

```
# verl/utils/megatron/router_replay_utils.py
# pp_gather 对 int16 路由的收集：NCCL 无 int16 数据类型，走位视图绕行。
if local_layers_router_map.is_nested:
    # 嵌套路径使用 all_gather_object，pickle 保留原 dtype，无需位视图。
    local_layers_router_map = local_layers_router_map.to("cpu")
    layers_topk_idx_global_list = [None] * world_size
    torch.distributed.all_gather_object(layers_topk_idx_global_list, local_layers_router_map, pp_
    group)
else:
    # int16 在 NCCL 上无对应数据类型，直接 all_gather 会失败。这里先把
    # 张量按字节 reinterpret 成 uint8（元素数翻倍、字节数不变），收集后再
    # view 回 int16，保证跨 PP rank 拼接前数值与 dtype 都正确。
    payload = local_layers_router_map.to(device_name).contiguous().view(torch.uint8)
    layers_topk_idx_global_list = [torch.empty_like(payload) for _ in range(world_size)]
    torch.distributed.all_gather(
        tensor=payload,
        tensor_list=layers_topk_idx_global_list,
        group=pp_group,
        async_op=False,
    )
    layers_topk_idx_global_list = [t.view(torch.int16) for t in layers_topk_idx_global_list]
```

评论区精华

该 PR 无 review 评论（comments_count=0、review_comments=0），但从提交信息与代码注释可以反推设计决策要点：

> int16 has no NCCL datatype, so the collective must see the uint8 view while the recorded routes land back as int16. —— 测试注释明确记录了通信层约束与绕行方案，这是本 PR 最核心的技术取舍。
>
> A model that hooks a subset -- e.g. DeepSeek-V4, whose first three layers are hash-routed -- silently shifts every layer's target unless its skipped layers are hooked too. —— 模块 docstring 说明了新增 num_fired/num_targets 契约校验的必要性，避免部分层未接入 hook 时静默错位。

另一个隐含决策是移除 replay_mask：控制器用 all-zero 目标行的 duplicate-top-k 检测替代显式 mask，从而简化状态机，但代价是 topk=1 模型被引擎显式拒绝（该检查无法触发）。

- 暂无高价值评论线程

风险与影响

- 风险：

1. 跨模块数据契约变更：int16 替换 uint8 涉及 padding、engine、model 工具层，仍可能有残留的 uint8 假设（如自定义 rollout 后端、checkpoint 或 reward 侧），一旦遗漏会出现 dtype 不匹配或静默截断。
2. NCCL 位视图依赖：pp_gather 与 merge_router_topk_indices 依赖 uint8 view 传输后再还原，要求两端字节序与内存布局一致；postprocess_thd_engine 的 dtype 透传是配套修复，但 CP 拼接、动态 batch restore 等其他路径仍需关注。
3. 行为变更：REPLAY 后 num_fired != num_targets 从静默容忍变为直接 raise；对只 hook 部分 MoE 层的模型（如 DeepSeek-V4 前 3 层 hash-routed）要求显式补 hook，否则会报错而非错位。
4. 测试覆盖局限：新增 / 更新测试全部为 CPU 单测，缺少 GPU/ 多卡端到端验证；multi-rank SP 与真实 SparseMoeBlock 前向仍由 e2e shell 脚本覆盖。 - 影响：对用户而言，支持超过 256 个专家的 MoE 模型在 Megatron 与 VeOmni 上的路由记录 / 回放不再被截断，同时路由索引内存相对 int64 减少约 75%。对系统而言，VeOmni router replay 的 controller-engine 职责边界更清晰，后续新增微批编排不再需要 side-channel；Megatron 与 VeOmni 两侧的路由 dtype 行为对齐。对团队而言，重构减少了约 300 行维护代码，但新集成方必须遵循每 micro-batch 的生命周期（begin_microbatch -> 前向 -> take_recorded/clear）。 - 风险标记：核心数据契约跨模块变更，NCCL 无 int16 依赖位视图，REPLAY 契约断言行为变更，缺少 GPU 端到端测试

关联脉络

- PR #7340 [megatron] fix: bugfix qwen 3 qwen 3.5 router replay: 同一 router replay 功能线，修复 Megatron 侧路由回放静默失效问题。
- PR #7297 [megatron] fix: make DeepSeek-V4 context parallelism actually runnable: DeepSeek-V4 在 Megatron 上的支持线，本 PR 的 int16 改造正是为 DeepSeek-V4 等超大 MoE 服务。
- PR #6804 [BREAKING][rollout] feat: Add Multimodal Continuous Token: 涉及 DeepSeek-V4 ContinuousToken 与 agent_loop 相关改动，与本次 tokenizer builder 修复同源。
- PR #7358 [megatron] feat: bucket packed sequence lengths: 涉及 Megatron preprocess/postprocess thd_engine 与 mcore/util.py，与本 PR 的 dtype 透传改动在同一模块。