PR #6432 完整报告

verl-project/verl

[megatron,rollout] fix: align MTP loss and rollout metrics

合并时间：2026-05-25 11:48

原文链接：http://prhub.com.cn/verl-project/verl/pull/6432

执行摘要

- 一句话：修复 Megatron MTP 训练与 rollout 指标对齐
- 推荐动作：该 PR 修复了多个 MTP 训练关键 bug（loss 污染、mask 错位、recompute 崩溃），建议优先合并。但需关注 bias 处理问题，确认目标模型输出层是否使用 bias；若使用，后续应跟踪补充 bias 加法逻辑。代码结构清晰，但测试覆盖不足是主要风险。建议团队内部补充 CI 测试用例。

功能与动机

原始 MTP loss 计算通过 ColumnParallelLinear 直接传递梯度到 lm_head，导致验证 / 基础模型分布偏移；loss mask 通过通用 nested tensor 转换，错将 prompt 位置标记为有效 MTP 位置；activation recompute 因非 tensor 元数据 PackedSeqParams 传入 checkpoint wrapper 而崩溃；reference 模型因 HF 配置加载 MTP 块浪费显存且导致 labels.clone() on None 失败。详见 PR body Bug Background。

实现拆解

1. MTP loss 梯度隔离（verl/models/mcore/mtp_patch.py）：在 Legacy API _megatron_gptmodel_postprocess 中，将 compute_output_layer_and_language_model_loss 替换为 torch.func.functional_call，对 output_layer.named_parameters() 和 output_weight 执行 detach()，确保 MTP 辅助 loss 不会更新 lm_head。
2. MTP loss mask 对齐（verl/models/mcore/model_forward.py）：新增 _build_mtp_loss_mask_nested 函数，根据 input_ids_lengths 将 response mask 扩展为 [prompt zeros; response mask] 格式，与 packed input_ids 对齐；在 gptmodel_forward_model_engine 中 MTP 模式下调用此函数替代通用 nested tensor 转换。
3. Activation recompute 修复（verl/models/mcore/mtp_patch.py）：新增 patch_mtp_layer_checkpointed_forward 和 unpatch_mtp_layer_checkpointed_forward，通过 monkey patch MultiTokenPredictionLayer._checkpointed_forward，使 checkpoint wrapper 只保存 tensor 输入，非 tensor 参数在 recompute closure 内恢复。
4. Reference 模型 MTP 禁用（verl/workers/engine/megatron/transformer_impl.py）：在 forward_only 且 mtp_num_layers=0 时，对 ref 模型调用 patch_postprocess 覆盖 MTP 后处理，避免加载未使用的 MTP 层，同时防止 labels=None 的异常。

5. Speculative decoding metric 聚合（verl/trainer/ppo/ray_trainer.py、verl/workers/rollout/vllm_rollout/vllm_async_server.py、verl/workers/rollout/sglang_rollout/async_sglang_server.py）：新增 compute_spec_decode_metrics 函数，按 per-request 宏平均聚合 draft/accept/verify 统计；在 vLLM 和 sglang 的 rollout 后端收集原始数据；在 PPO trainer 的 fit 循环中集成 metric 更新。
6. vLLM MTP drafter 权重同步（verl/workers/rollout/vllm_rollout/utils.py）：添加 _get_drafter_model、_use_mtp_drafter_weight_sync、_iter_all_models 等辅助方法，使 update_weights_from_ipc、monkey_patch_model 等函数能同时作用于主模型和 MTP drafter；仅在 speculative method 为 "mtp" 时同步 drafter 权重。

关键文件：

- verl/models/mcore/mtp_patch.py（模块 模型层；类别 source；类型 data-contract；符号 patch_mtp_layer_checkpointed_forward, unpatch_mtp_layer_checkpointed_forward, _patched_checkpointed_forward, run）：核心修改：MTP loss 梯度隔离、activation recompute patch，影响训练正确性和稳定性
- verl/models/mcore/model_forward.py（模块 模型层；类别 source；类型 data-contract；符号 _build_mtp_loss_mask_nested）：新增 _build_mtp_loss_mask_nested 函数，修正 MTP loss mask 对齐
- verl/workers/rollout/vllm_rollout/utils.py（模块 采样层；类别 source；类型 core-logic；符号 _get_drafter_model, _get_draft_model_config, _use_mtp_drafter_weight_sync, _iter_all_models）：添加 drafter weight sync 支持，使 MTP drafter 也能接收 actor 权重
- verl/trainer/ppo/ray_trainer.py（模块 训练器；类别 source；类型 core-logic；符号 compute_spec_decode_metrics）：新增 spec decode metrics 函数并集成到训练流程
- verl/workers/engine/megatron/transformer_impl.py（模块 引擎层；类别 source；类型 dependency-wiring）：禁用 reference 模型 MTP 层，修复内存和异常
- verl/utils/vllm/vllm_fp8_utils.py（模块 工具层；类别 source；类型 core-logic；符号 load_quanted_weights）：支持 FP8 量化模式下 MTP drafter 权重加载
- examples/mtp_trainer/run_mimo_7b_mtp_rl_vllm_sgl_megatron.sh（模块 示例脚本；类别 other；类型 core-logic）：新增 MTP 训练启动脚本，为复现实验提供配置

关键符号：compute_spec_decode_metrics, _build_mtp_loss_mask_nested, patch_mtp_layer_checkpointed_forward, _patched_checkpointed_forward, load_quanted_weights, _get_drafter_model, _use_mtp_drafter_weight_sync, _iter_all_models

关键源码片段

verl/models/mcore/mtp_patch.py

核心修改：MTP loss 梯度隔离、activation recompute patch，影响训练正确性和稳定性

```
# 在 GPTModel 的 _megatron_gptmodel_postprocess 函数中，
# 当 mtp_num_layers > 0 且 labels 不为 None 且未使用新版 process_mtp_loss API 时，
```

```
# 进入旧版 Manual Rolling 分支。此段展示 MTP loss 计算的关键修改:
else:
    mtp_labels = labels.clone()
    hidden_states_list = torch.chunk(hidden_states, 1 + self.config.mtp_num_layers, dim=0)
    hidden_states = hidden_states_list[0]
    if loss_mask is None:
        loss_mask = torch.ones_like(mtp_labels)
    cp_group = getattr(self, "cp_group", None)
    for mtp_layer_number in range(self.config.mtp_num_layers):
        mtp_labels, _ = roll_tensor(
            mtp_labels, shifts=-1, dims=-1,
            cp_group=cp_group, packed_seq_params=packed_seq_params,
        )
        loss_mask, num_tokens = roll_tensor(
            loss_mask, shifts=-1, dims=-1,
            cp_group=cp_group, packed_seq_params=packed_seq_params,
        )
        # Detach output-layer params so MTP loss does not update lm_head.
        output_layer_params = {k: v.detach() for k, v in self.output_layer.named_parameters()}
        output_layer_buffers = dict(self.output_layer.named_buffers())
        # 使用 functional_call 计算 logits，参数都是 detach 状态，不会产生梯度到 lm_head
        mtp_logits, _ = torch.func.functional_call(
            self.output_layer,
            {**output_layer_params, **output_layer_buffers},
            args=(hidden_states_list[mtp_layer_number + 1],),
            kwargs={
                "weight": output_weight.detach() if output_weight is not None else None,
                "runtime_gather_output": runtime_gather_output,
            },
        )
        mtp_loss = self.compute_language_model_loss(mtp_labels, mtp_logits)
        mtp_loss = loss_mask * mtp_loss
        if self.training:
            MTPLossLoggingHelper.save_loss_to_tracker(
                mtp_loss, ... # 省略细节
            )
```

verl/models/mcore/model_forward.py

新增 _build_mtp_loss_mask_nested 函数，修正 MTP loss mask 对齐

```
def _build_mtp_loss_mask_nested(response_mask, input_ids_lengths, response_attention_mask):
    """Build a nested loss_mask aligned to ``input_ids = [prompt; response]`` for MTP.

    ``response_mask`` is response-only data. This expands it to full packed
    input length as prompt zeros followed by valid response positions.
    """
    if isinstance(response_mask, NestedTensor):
        response_offsets = response_mask.offsets().tolist()
        response_lengths = [response_offsets[i + 1] - response_offsets[i] for i in
```

```
            range(len(response_offsets) - 1)]
            batch_size = len(response_lengths)
            response_values = response_mask.values()
        else:
            # 对于 padded 格式，从 response_attention_mask 提取每个样本的有效 response 长度
            assert response_attention_mask is not None, "response_attention_mask is required"
            assert not isinstance(response_attention_mask, NestedTensor)
            assert response_attention_mask.shape == response_mask.shape
            batch_size = response_mask.shape[0]
            response_lengths = response_attention_mask.to(torch.int32).sum(dim=-1).tolist()

        assert len(input_ids_lengths) == batch_size

        pieces = []
        for i in range(batch_size):
            actual_total = int(input_ids_lengths[i])
            actual_response = int(response_lengths[i])
            actual_prompt = actual_total - actual_response
            assert actual_prompt >= 0
            # 构建 [prompt zeros; response mask] 的 loss_mask
            prompt_pad = torch.zeros(actual_prompt, dtype=response_mask.dtype, device=response_
            mask.device)
            if isinstance(response_mask, NestedTensor):
                response_piece = response_values[response_offsets[i] : response_offsets[i + 1]]
            else:
                response_piece = response_mask[i, :actual_response]
            full = torch.cat([prompt_pad, response_piece], dim=0)
            assert full.shape[0] == actual_total
            pieces.append(full)

        return torch.nested.nested_tensor(pieces, layout=torch.jagged)
```

verl/workers/rollout/vllm_rollout/utils.py

添加 drafter weight sync 支持，使 MTP drafter 也能接收 actor 权重

```
class VllmWorker:
    # ... 现有方法 ...

    def _get_drafter_model(self):
        """Return the drafter's model object, or None if unavailable."""
        drafter = getattr(self.model_runner, "drafter", None)
        return drafter.model if drafter is not None and hasattr(drafter, "model") else None

    def _get_draft_model_config(self):
        """Return the draft model config from speculative_config, or None."""
        spec = self.model_runner.vllm_config.speculative_config
        return spec.draft_model_config if spec is not None and spec.draft_model_config is not
        None else None

    def _use_mtp_drafter_weight_sync(self):
```

```
        """Return whether the vLLM MTP drafter should receive actor weights."""
        spec = self.model_runner.vllm_config.speculative_config
        # 仅当 speculative method 为 mtp 且 drafter 存在时才同步
        return spec is not None and spec.method == "mtp" and self._get_drafter_model() is not
        None

    def _iter_all_models(self):
        """Yield models that need weight updates.
        Only vLLM MTP drafter sync is supported for now.
        """
        yield self.model_runner.model
        if self._use_mtp_drafter_weight_sync():
            yield self._get_drafter_model()

    def _iter_all_models_with_config(self):
        """Yield (model, model_config) for models that need post-processing."""
        yield self.model_runner.model, self.model_runner.vllm_config.model_config
        if self._use_mtp_drafter_weight_sync():
            draft_cfg = self._get_draft_model_config()
            if draft_cfg is not None:
                yield self._get_drafter_model(), draft_cfg
```

评论区精华

在代码审查中，gemini-code-assist[bot] 指出 functional_call 中未处理输出层 bias：ColumnParallelLinear 可能返回 (logits, bias)，忽略 bias 会导致 MTP loss 计算错误。建议修正 bias 加法并优化参数字典初始化。该评论为 critical 级别，但合并者直接批准，未要求额外修改，问题在当前 PR 中仍存在。

- MTP loss functional_call 忽略 bias 可能导致错误 (correctness): PR 合并时未修改该问题，合并者直接批准。问题仍存在，需要后续跟进。

风险与影响

- 风险： 1) MTP loss 梯度隔离依赖 functional_call，但未处理输出层 bias（如 review 指出的），若模型使用 bias 则 MTP auxiliary loss 错误。 2) MTP loss mask 对齐函数改变原数据流，可能影响非 packed 场景的兼容性。 3) Activation recompute patch 为 monkey patch，可能与其他自定义 checkpoint wrapper 冲突。 4) Reference 模型 MTP 禁用通过 patch_postprocess 覆盖，未来 Megatron 版本如果调整 MTP 接口可能导致不兼容。 5) Spec decode metrics 聚合函数为新增代码，未经大规模验证，可能边界异常。
  - 影响：直接使用 Megatron MTP 训练的团队需要更新代码以应用此修复；vLLM/sglang speculative decoding 用户将获得正确的 accept rate 和 accept length metric。影响范围主要为训练代码和 rollout metric，无 API breaking change。建议用户配合使用新增的 run_mimo_7b_mtp_rl_vllm_sgl_megatron.sh 脚本进行验证。
  - 风险标记：核心路径变更，缺少测试覆盖，Bias 未处理

关联脉络

- 暂无明显关联 PR