PR #6699 完整报告

verl-project/verl

[fsdp, trainer] fix: detach model_output and loss metrics to stop per-micro-batch graph retention in actor update

合并时间：2026-06-12 12:06

原文链接：http://prhub.com.cn/verl-project/verl/pull/6699

执行摘要

- 一句话：修复 actor 更新中 per-micro-batch graph retention 导致的内存泄漏
- 推荐动作：建议读者精读本 PR 的 root cause 分析（Issue #6698）和回归测试设计，了解训练引擎中 per-micro-batch 梯度图保留的常见陷阱。修复手法（early detach 非必要 tensor）是此类问题的标准解法。测试中的 _TinyCheckpointedLM 和 weakref 验证模式值得在类似场景复用。

功能与动机

Issue #6698 详细报告了 verl/workers/engine/fsdp/transformer_impl.py 中 forward_step 返回的 model_output 携带 grad_fn，被 forward_backward_batch 收集后导致整个训练过程的自动求导图无法释放，在 LoRA + 长序列场景下 OOM 的训练中断。PR 直接修复该问题的根因，避免每微批次 0.27 GiB 显存泄漏，使得 Qwen3-8B + LoRA rank 32 的多轮工具调用训练在第一个 actor update 就可完成且指标正常。

实现拆解

1. 核心逻辑修复（verl/workers/engine/fsdp/transformer_impl.pyforward_step 方法）：在 loss 和 metrics 计算完成后，model_output 被组装进返回字典 output 之前，使用字典推导式遍历 model_output 的所有值，若值为张量且具有 grad_fn（即有梯度信息），则调用 .detach() 获得副本。这样 output["model_output"] 中的张量不再持有计算图引用，但 loss 张量仍单独保留并用于 backward()。
2. 回归测试（新增 tests/workers/test_engine_forward_step_detach_on_cpu.py）：仿照真实场景设计了迷你模型 _TinyCheckpointedLM，包含冻结 embedding、需梯度的中间激活、checkpointed 可训练全连接层，并在 checkpoint 处保存弱引用。测试用例先执行 forward_step，验证返回的 model_output 中所有张量 grad_fn 为 None，然后在失去 loss 引用后确认 checkpoint 保存的输入已被垃圾回收，确保 output_lst 持有返回结果时不会阻碍内存释放。
3. 伴随调整（来自提交历史）：最初 ppo_loss 函数中也对指标张量做了 detach，后在测试验证中发现 Metric.append 已对标量执行 .detach().item()，该处修改冗余，已通过 revert 删除，保持 diff 最小。

关键文件：

- verl/workers/engine/fsdp/transformer_impl.py（模块 引擎；类别 source；类型 core-logic；符号 forward_step）：核心修复文件：在 forward_step 方法中构建返回字典前对 model_output 中的张量执行 detach，切断每个微批次的计算图引用，从根本上避免显存堆积。
- tests/workers/test_engine_forward_step_detach_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _TinyCheckpointedLM, init, forward, _make_engine_stub）：新增回归测试，精确复现了 PEFT + checkpoint 场景下的梯度图 retention 机制，验证 model_output 中的张量 grad_fn 为 None，以及 backward 后 checkpoint 保存的输入可被 GC 回收，确保修复有效且防止回归。

关键符号：forward_step, test_forward_step_output_carries_no_grad_fn_and_releases_graph

关键源码片段

verl/workers/engine/fsdp/transformer_impl.py

核心修复文件：在 forward_step 方法中构建返回字典前对 model_output 中的张量执行 detach，切断每个微批次的计算图引用，从根本上避免显存堆积。

```
# Detach model outputs before they are appended to forward_backward_batch's
# output_lst: they are only consumed for metrics/postprocessing after backward,
# and keeping their grad_fn alive retains part of every micro-batch's autograd
# graph until the whole batch finishes. With PEFT (enable_input_require_grads)
# this pins the checkpointed embedding output plus its gradient buffer per
# micro-batch (~2 x [total_nnz, hidden] for long sequences), which accumulates
# across micro-batches and OOMs the actor update.
# （以下代码位于 `forward_step` 方法中，`loss` 和 `metrics` 计算之后）
model_output = {
    key: value.detach() if torch.is_tensor(value) and value.grad_fn is not None else value
    for key, value in model_output.items()
}
output = {
    "model_output": model_output,
    "loss": loss.detach().item(),
    "metrics": metrics,
}
```

tests/workers/test_engine_forward_step_detach_on_cpu.py

新增回归测试，精确复现了 PEFT + checkpoint 场景下的梯度图 retention 机制，验证 model_output 中的张量 grad_fn 为 None，以及 backward 后 checkpoint 保存的输入可被 GC 回收，确保修复有效且防止回归。

```
class _TinyCheckpointedLM(torch.nn.Module):
    """Frozen embedding + checkpointed trainable block, mimicking a PEFT/LoRA
    base model with gradient checkpointing and enable_input_require_grads."""

    def __init__(self):
        super().__init__()
```

```
        self.embed = torch.nn.Embedding(VOCAB, HIDDEN)
        self.embed.weight.requires_grad_(False) # frozen base
        self.proj = torch.nn.Linear(HIDDEN, HIDDEN, bias=False) # the trainable part
        self.saved_block_input = None # weakref to the checkpoint-saved input

    def forward(self, input_ids=None, use_cache=False):
        x = self.embed(input_ids)
        # PEFT enable_input_require_grads: embedding output requires grad so
        # gradients can flow into trainable params under checkpointing.
        x.requires_grad_(True)
        hidden = torch.utils.checkpoint.checkpoint(self.proj, x, use_reentrant=False)
        self.saved_block_input = weakref.ref(x)
        return SimpleNamespace(hidden=hidden)


def test_forward_step_output_carries_no_grad_fn_and_releases_graph():
    eng = _make_engine_stub()
    micro_batch = TensorDict(
        {"input_ids": torch.randint(0, VOCAB, (1, SEQ))},
        batch_size=[1],
    )
    loss, output = FSDPEngineWithLMHead.forward_step(eng, micro_batch, _loss_fn, forward_
    only=False)
    # The live loss must still drive backward into the trainable params.
    assert loss.grad_fn is not None
    loss.backward()
    assert eng.module.proj.weight.grad is not None

    # Contract: nothing in the collected per-micro-batch output may keep the
    # autograd graph alive once the loss reference is dropped.
    for key, value in output["model_output"].items():
        assert not (torch.is_tensor(value) and value.grad_fn is not None), \
            f"model_output[{key!r}] still attached to the autograd graph"

    saved_input = eng.module.saved_block_input
    assert saved_input() is not None # sanity: alive while loss exists
    del loss
    gc.collect()
    # `output` is intentionally still held, like forward_backward_batch's
    # output_lst holds it across the remaining micro-batches of the batch.
    assert saved_input() is None, \
        "checkpoint-saved block input (embedding output) survived backward: the per-micro-batch
        output is retaining the autograd graph"
```

评论区精华

- 设计决策：作者在 PR body 提到，最初在 ppo_loss 中也分离了指标张量，但发现 Metric.append 已经对 scalar 张量执行 .detach().item()，因此 metrics 不会持有计算图，该处修改是冗余的，已 revert（commit 5ad19e8）。核心修复仅需在 forward_step 中

detach model_output，确保 diff 最小、聚焦根因。

- CI 失败分析：作者在评论中指出 CI 的 cpu_unit_tests 和 e2e_ppo_trainer_fsdp-qwen2_5vl-3b 失败均与本 PR 无关（cpu_unit_tests 的失败项 test_distillation_topk_symmetry_on_cpu 在 main 上已存在；e2e 失败是数据加载信号问题），且其他 e2e 工作流均通过，表明变更没有引起新回归。
  - 分离 ppo_loss 指标张量的必要性 (design): 只需在 forward_step 中分离 model_output，不需要修改 ppo_loss 的 metrics 部分，以减少 diff 和潜在混乱。

风险与影响

- 风险：
  - 回归风险低：变更仅 11 行新增（主文件），对 model_output 中的张量执行 .detach()，不改变任何控制流或数学计算逻辑。loss 张量仍保持计算图用于 backward()，被 detach 的张量只用于后续指标聚合和日志（只读），不影响训练正确性。新增的回归测试精确复现了关键路径，验证了 grad_fn 断开和内存释放。
  - 潜在风险：若将来有代码依赖 model_output 中张量的 grad_fn（例如在 micro-batch 结束后还想通过它访问某些统计信息），本 PR 可能会破坏这种隐式假设。但在当前代码库中，forward_backward_batch 收集的 output_lst 仅在训练循环的最后用于组装批次级输出，不涉及反向传播，因此风险极低。
  - 测试覆盖：CPU 回归测试覆盖了最核心的梯度隔离与内存释放场景，但未覆盖多 GPU 或混合精度场景；不过 detach 操作是设备无关且类型无关的，跨平台行为一致。
  - 影响：影响范围：所有使用 FSDP/FSDP2 后端训练 actor 模型（包括 PPO、GRPO）的用户，尤其是结合 LoRA 或多轮工具调用长序列训练的场景。无需修改任何 API 或配置，修复对用户透明。影响程度：对受影响的训练任务（之前 OOM）属于 critical 修复；对无明显内存压力的任务，detach 操作的开销极小，性能无退化。此外，修复同样适用于 FSDPEngineWithValueHead（critic 模型），因为其继承自被修改的类，也会受益于 model_output 的 detach。
  - 风险标记：核心路径变更

关联脉络

- PR #6698 Per-micro-batch GPU memory leak in engine actor update: model_output/metrics retain autograd graph (OOM with LoRA + long sequences): 该 Issue 提供了详尽的问题描述、root cause 分析和内存走势图，是此 PR 的直接动机和修复依据。