

PR #6952 完整报告

verl-project/verl

[megatron] fix: free Transformer-Engine FP8 weight workspaces on CPU offload

合并时间: 2026-07-07 11:00

原文链接: <http://prhub.com.cn/verl-project/verl/pull/6952>

执行摘要

- 一句话: 修复 Megatron CPU offload 未释放 TE FP8 工作空间的问题
- 推荐动作: 建议合入。该 PR 解决了 Megatron FP8 训练中一个关键的内存泄漏问题, 改动精炼且已通过真实环境验证。值得关注其设计: `_clear_te_fp8_weight_workspaces` 的模块遍历 + 属性检查模式, 可作为类似 TE 缓存清理的参考。

功能与动机

PR body 明确指出: `offload_megatron_model_to_cpu` 从未清理 Transformer-Engine 的 FP8 权重工作空间, 导致原生 FP8 检查点的量化权重副本 (`Float8Tensor/Float8BlockwiseQTensor`) 在 offload 后仍驻留 GPU, 与 rollout 引擎的权重重载冲突, 造成 OOM。

实现拆解

1. 新增辅助函数 `_clear_te_fp8_weight_workspaces` (`verl/utils/megatron_utils.py`) :
 - 遍历模型的所有子模块, 通过 `getattr(submodule, '_fp8_workspaces', None)` 安全获取可能存在的 FP8 工作空间字典。
 - 若字典非空, 调用 `workspaces.clear()` 释放其内容, 并累计清理条目数。
 - 该函数仅在 FP8 路径生效, 在 bf16/fp16 模型中是无操作。
2. 在 `offload_megatron_model_to_cpu` 末尾调用:
 - 对每个模型块, 在参数 offload 完成后、`gc.collect()` 和 `empty_cache()` 之前, 调用 `_clear_te_fp8_weight_workspaces` 并记录日志。
3. 代码优化 (根据 Review 反馈) :
 - 移除对 `model_chunk.modules()` 的防御性 `getattr` 检查, 因为 `model_chunk` 始终是 `nn.Module`, 直接调用。
 - 保留对 `_fp8_workspaces` 的 `getattr` 检查, 因为该属性仅存在于 TE FP8 层。

关键文件:

- `verl/utils/megatron_utils.py` (模块 工具函数; 类别 source; 类型 core-logic; 符号 `_clear_te_fp8_weight_workspaces`): 本次变更的唯一文件, 新增 `_clear_te_fp8_weight_workspaces` 函数并在 `offload_megatron_model_to_cpu` 中调用, 修复 FP8 权重工作空间的内存泄漏。

关键符号: `_clear_te_fp8_weight_workspaces`, `offload_megatron_model_to_cpu`

关键源码片段

verl/utils/megatron_utils.py

本次变更的唯一文件，新增 `_clear_te_fp8_weight_workspaces` 函数并在 `offload_megatron_model_to_cpu` 中调用，修复 FP8 权重工作空间的内存泄漏。

```
def _clear_te_fp8_weight_workspaces(model_chunk): """Clear cached
Transformer-Engine FP8 weight workspaces on a model chunk. When training with an
FP8 param dtype (or loading a native-FP8 checkpoint), Transformer-Engine ``Linear``
/ ``GroupedLinear`` layers cache quantized (``Float8Tensor`` /
``Float8BlockwiseQTensor``) copies of their weights in ``module._fp8_workspaces``
(keyed ``weight``, ``weight0..weightN``). These caches are plain tensors, not
``nn.Parameter``s or registered buffers, so the parameter/buffer offloading in :
func:`offload_megatron_model_to_cpu` never frees them and they stay resident on the
GPU -- for large MoE checkpoints this can be tens of GiB per rank that defeats the
offload. Transformer-Engine lazily rebuilds the workspace on the next forward, so
dropping it here is safe. This is a no-op for bf16/fp16 models (the attribute is absent
or empty). Returns the number of cached workspace entries cleared. """
    cleared = 0
    # model_chunk is always an nn.Module (or DDP wrapper); .modules() recurses #
    through wrappers automatically for submodule in model_chunk.modules(): #
    _fp8_workspaces only exists on TE Linear/GroupedLinear layers; # safe getattr
    avoids crashes on other submodules workspaces = getattr(submodule, "
_fp8_workspaces", None) if isinstance(workspaces, dict) and workspaces:
        cleared += len(workspaces) workspaces.clear() # release the quantized
weight tensors
    return cleared
def offload_megatron_model_to_cpu(models): ...
for model_chunk in models: ... # existing parameter/buffer offload logic #
Drop TE FP8 weight workspace caches after parameter offload cleared =
_clear_te_fp8_weight_workspaces(model_chunk) if cleared:
    logger.debug("Cleared%dTE FP8 weight workspaces on offload", cleared) gc.collect()
    get_torch_device().empty_cache()
```

评论区精华

Review 中 `gemini-code-assist[bot]` 指出初始实现存在过度防御性编程：对 `model_chunk.modules()` 使用了冗余的 `getattr` 和 `None` 检查。作者随后修正，直接调用 `model_chunk.modules()`，同时保留了必要的 `getattr` 检查（因为 `_fp8_workspaces` 确实是可选属性）。

- 防御性编程 vs 简洁性：`model_chunk.modules()` 是否需要 `getattr` 保护 (style)：作者采纳建议，直接调用 `model_chunk.modules()`，同时保留对 `_fp8_workspaces` 的 `getattr` 检查（该属性确实可选）。

风险与影响

- 风险：

- 回归风险低：仅在 FP8 路径生效，且 TE 会在下一次 forward 时惰性重建工作空间。
- 兼容性：对 bf16/fp16 模型完全无影响。
- 测试覆盖不足：由于需要 GPU + TE + FP8 模型，无法在 CPU CI 中测试，但已在真实环境验证。
- 影响：
 - 用户：使用 Megatron + FP8 训练的用户将大幅减少显存泄漏，避免 colocated hybrid training 中的 OOM。
 - 系统：每 rank 显存节省数十 GiB，显著提升大规模 FP8 模型的训练稳定性。
 - 团队：改动极小 (+31/-0)，无新增依赖，风险极低。
 - 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #6901 [megatron, perf] feat: pad BSHD micro-batches to mini-batch max seq_len: 同为 Megatron 性能 / 内存优化相关，且改动了同文件 verl/utils/megatron_utils.py 的相邻区域。