

PR #6635 完整报告

verl-project/verl

[megatron] fix: Fix Megatron optimizer offloading when `use\_precision\_aware\_optimizer=True`

合并时间: 2026-06-08 11:26

原文链接: <http://prhub.com.cn/verl-project/verl/pull/6635>

执行摘要

- 一句话: 修复 Megatron 精度感知优化器的 offload 遗漏
- 推荐动作: 这是一个针对特定优化器选项的精准修复, 代码改动小但有明确价值。对于需要 GPU 内存优化的 Megatron 用户, 建议升级并启用该功能。设计上简单直接, 适合作为了解 Megatron 优化器状态结构的参考案例。

功能与动机

根据 PR body, 启用 `use_precision_aware_optimizer` 后, 优化器状态中的 `master_param` 未被 offload, 导致训练时 GPU 内存超限 (OOM)。该修复使得 Megatron 后端可以正常使用精度感知优化器。

实现拆解

1. 修改 offload 函数: 在 `verl/utils/megatron_utils.py` 的 `offload_megatron_optimizer` 函数中, 遍历 optimizer state 时, 除已有的 `exp_avg/exp_avg_sq` 外, 增加对 `master_param` 键的检查, 将其搬运到 CPU (`non_blocking=True`)。
2. 修改 load 函数: 对应的 `load_megatron_optimizer` 函数中, 同样在搬运 `exp_avg/exp_avg_sq` 后, 增加对 `master_param` 的搬运回 GPU (`get_device_id()`, `non_blocking=True`)。
3. 新增单元测试: 在 `tests/utils/megatron/test_optimizer_offload_and_load.py` 中, 构建一个小型 GPT 模型并初始化精度感知优化器, 调用 offload 后验证所有 optimizer state (包括 `master_param`) 已转移至 CPU, 调用 load 后验证状态回到 GPU。测试包括分布式环境初始化和清理 (按 review 建议调整了销毁顺序)。
4. 配置与使用: 通过添加 `actor_rollout_ref.actor.optim.override_optimizer_config.use_precision_aware_optimizer=True` 启用该功能。

关键文件:

- `verl/utils/megatron_utils.py` (模块 优化器; 类别 source; 类型 core-logic; 符号 `offload_megatron_optimizer`, `load_megatron_optimizer`): 核心源码文件, 修改了优化器 offload/load 逻辑, 新增对 `master_param` 的搬运处理。
- `tests/utils/megatron/test_optimizer_offload_and_load.py` (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `init_model`, `init_precision_aware_optimizer`, `precision_aware_optimizer_is_on_device`, `test_precision_aware_optimizer_offload_and`

_load)：新增的单元测试文件，验证优化器 offload/load 的正确性，包含精度感知优化器的特殊检查。

关键符号：offload_megatron_optimizer, load_megatron_optimizer, init_model, init_precision_aware_optimizer, precision_aware_optimizer_is_on_device, test_precision_aware_optimizer_offload_and_load

关键源码片段

verl/utils/megatron_utils.py

核心源码文件，修改了优化器 offload/load 逻辑，新增对 `master_param` 的搬运处理。

```
# offload_megatron_optimizer 节选 (verl/utils/megatron_utils.py)
# 在非 HybridDeviceOptimizer 的分支中，增加 master_param 处理
opt_state_dict_values = _opt.optimizer.state.values()
for v in opt_state_dict_values:
    if "exp_avg" in v:
        v["exp_avg"] = v["exp_avg"].to("cpu", non_blocking=True)
    if "exp_avg_sq" in v:
        v["exp_avg_sq"] = v["exp_avg_sq"].to("cpu", non_blocking=True)
    # 新增：当 use_precision_aware_optimizer=True 时，
    # master_param 存储了 32-bit master 权重，也必须 offload 到 CPU
    if "master_param" in v:
        v["master_param"] = v["master_param"].to("cpu", non_blocking=True)

# load_megatron_optimizer 节选 (对称增加)
opt_state_dict_values = _opt.optimizer.state.values()
for v in opt_state_dict_values:
    if "exp_avg" in v:
        v["exp_avg"] = v["exp_avg"].to(get_device_id(), non_blocking=True)
    if "exp_avg_sq" in v:
        v["exp_avg_sq"] = v["exp_avg_sq"].to(get_device_id(), non_blocking=True)
    # 新增：将 master_param 搬回 GPU
    if "master_param" in v:
        v["master_param"] = v["master_param"].to(get_device_id(), non_blocking=True)
```

tests/utils/megatron/test_optimizer_offload_and_load.py

新增的单元测试文件，验证优化器 offload/load 的正确性，包含精度感知优化器的特殊检查。

```
def test_precision_aware_optimizer_offload_and_load(tmp_path):
    # 初始化分布式环境 (简化后的单卡版本)
    torch.cuda.set_device(0)
    dist.init_process_group(
        backend="cpu:gloo,cuda:nccl",
        init_method=f"file://{tmp_path / 'rdzv_optimizer'}",
        rank=0,
        world_size=1,
    )
    mpu.initialize_model_parallel()
```

```

model_parallel_cuda_manual_seed(123)

try:
    # 初始化模型和优化器
    model = init_model() # 2 层 GPT, hidden=512, 使用 DistributedDataParallel
    optimizer = init_precision_aware_optimizer(model) # use_precision_aware_optimizer=True

    # 执行 step() 触发 optimizer 内部状态初始化
    for chunk in model:
        chunk.zero_grad_buffer()
        optimizer.zero_grad()
        update_successful, _, _ = optimizer.step()
        assert update_successful

    # 验证初始 master_param 在 GPU 上且 shard_fp32_from_float16_groups 为 None
    assert precision_aware_optimizer_is_on_device(optimizer, torch.device("cuda"))

    # offload 后所有状态 (含 master_param) 应在 CPU
    offload_megatron_optimizer(optimizer)
    assert precision_aware_optimizer_is_on_device(optimizer, torch.device("cpu"))

    # load 后所有状态应回到 GPU
    load_megatron_optimizer(optimizer)
    assert precision_aware_optimizer_is_on_device(optimizer, torch.device("cuda"))
finally:
    mpu.destroy_model_parallel() # 必须在线程组销毁前调用
    dist.destroy_process_group()

```

评论区精华

Review 中 [gemini-code-assist\[bot\]](#) 指出测试 fixture 的 teardown 顺序问题：应先调用 `mpu.destroy_model_parallel()` 再 `dist.destroy_process_group()`，否则可能导致清理错误。该建议被采纳并在后续提交中修复。此外，[kaixih](#) 和 [wuxibin89](#) 审批通过。

- Test teardown 顺序：应先销毁模型并行状态再销毁进程组 (correctness)：作者采纳建议，调整了销毁顺序。

风险与影响

- 风险：本 PR 仅修改 `offload_megatron_optimizer` 和 `load_megatron_optimizer` 两个函数，影响范围限定在 Megatron 优化器 offload/load 路径。风险较低，但需注意：
 - 若用户未启用精度感知优化器，`master_param` 不存在，条件判断不会执行，无影响。
 - 新增的 `non_blocking` 拷贝与已有代码保持一致。
 - 测试覆盖了单卡场景，但多卡 / 分布式场景下需确保 `get_device_id()` 能正确返回本地 GPU。
 - 若其他代码直接操作 optimizer state (如 checkpoint 保存)，可能也需要同步更新，但本 PR 未涉及。

- 影响：影响范围：使用 Megatron 后端且启用 use_precision_aware_optimizer 的用户。
影响程度：修复了一个导致 OOM 的 bug，使该功能变得可用。兼容性：向后兼容，未修改 API 或配置文件。新增单元测试提升了回归防护。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR