

PR #3100 完整报告

radixark/miles

Support `--colocate-memory-peak-device gpu` under LoRA

合并时间: 2026-09-09 07:11

原文链接: <http://prhub.com.cn/radixark/miles/pull/3100>

执行摘要

- 一句话: 新增 `--colocate-memory-peak-device gpu` 对 LoRA 的支持, 通过管理梯度缓冲区生命周期优化内存占用。
- 推荐动作: 本 PR 值得精读。它解决了一个具体的、影响生产部署的内存优化问题, 实现方案聚焦且测试充分。主要关注点在于 `actor.py` 中 `sleep/wake_up` 的新控制流, 以及 `train.py` 中内存释放时序的调整。对于维护或扩展分布式训练内存管理模块的工程师, 这是很好的参考实现。

功能与动机

解决 `--colocate-memory-peak-device gpu` 模式下, 由于 LoRA 训练器持有梯度缓冲区而推理引擎需要恢复权重, 导致内存峰值无法有效降低的问题。PR body 明确指出: 'gpu mode resumes the engine weights before the trainer sleeps, and the trainer has to give back what it can first... LoRA also holds its grad buffers in the no-backup grad_buffer region... This adds that step and lifts the restriction.'

实现拆解

1. 核心逻辑增强 (`actor.py`): 在 `MegatronTrainRayActor` 中新增 `_grad_buffer_paused` 状态标志, 并实现 `offload_grad_buffer()` 方法。该方法调用 `torch_memory_saver.pause(tag="grad_buffer")` 暂停 LoRA 梯度缓冲区, 并更新状态以避免重复暂停。同时, 重构了 `sleep()` 和 `wake_up()` 方法, 使其在 LoRA 模式下分别处理 "grad_buffer" 和 "default" 两个内存区域的暂停与恢复。
2. 调度层接入 (`group.py`): 在 `miles/ray/train/group.py` 的 `TrainGroup` 中新增 `offload_grad_buffer` 异步方法, 将调用转发到所有活跃的 actor 单元, 使其在分布式训练中可用。
3. 训练流程编排 (`train.py`): 移除了 LoRA 对 GPU 模式的硬性限制。在 gpu 模式的执行循环中, 当使用 LoRA 时, 在 `clear_memory()` 之后、推理引擎加载权重之前, 显式调用 `actor_model.offload_grad_buffer()`。这调整了内存释放的时序, 确保推理引擎能优先获得所需内存。
4. 测试与模拟 (`test_shared_ppo_lifecycle.py`, `dummy_actor.py`): 新增了两个专门针对 LoRA 生命周期的单元测试, 验证 `sleep/wake_up` 的正确标签序列, 以及 `offload_grad_buffer` 与 `sleep` 的幂等性交交互。同时, 在测试用的 `DummyActor` 中补充了 `offload_grad_buffer` 方法模拟, 以支持相关测试。

关键文件：

- miles/backends/megatron_utils/actor.py（模块 训练 Actor；类别 source；类型 core-logic；符号 offload_grad_buffer, sleep, wake_up）：核心变更文件，实现了 LoRA 梯度缓冲区的主动卸载和生命周期管理，是解决内存峰值问题的关键。
- train.py（模块 训练循环；类别 source；类型 entrypoint）：训练主循环文件，调整了 gpu 模式下的内存释放时序，并移除了对 LoRA 的硬性限制，是功能启用的入口。
- miles/ray/train/group.py（模块 训练组管理；类别 source；类型 core-logic；符号 offload_grad_buffer）：分布式训练组管理文件，新增 offload_grad_buffer 异步方法，将核心变更接入分布式调度层。
- tests/fast/backends/megatron_utils/test_shared_ppo_lifecycle.py（模块 生命周期测试；类别 test；类型 test-coverage；符号 test_lora_sleep Pauses the grad_buffers and wake_up resumes them, test_offload_grad_buffer frees them once and sleep does not pause them again）：核心测试文件，新增了验证 LoRA 梯度缓冲区生命周期管理的单元测试，确保了实现的正确性。
- tests/fast/ray/train/dummy_actor.py（模块 测试 Actor；类别 test；类型 test-coverage；符号 offload_grad_buffer）：测试辅助文件，为模拟 actor 补充了 offload_grad_buffer 方法，以支持新的训练流程测试。

关键符号：offload_grad_buffer, sleep, wake_up

关键源码片段

miles/backends/megatron_utils/actor.py

核心变更文件，实现了 LoRA 梯度缓冲区的主动卸载和生命周期管理，是解决内存峰值问题的关键。

```
# 重构后的 sleep() 方法，新增对 LoRA 梯度缓冲区的管理
@with_logs
@timer
def sleep(self) -> None:
    assert self.args.offload_train
    if self._asleep:
        logger.info("sleep() called while already offloaded; skipping")
        return

    self._clear_quantized_weight_workspaces()
    clear_memory(clear_host_memory=True)
    print_memory("before offload model")
    should_log_cpu_memory = is_first_replica_megatron_main_rank() and hasattr(self, "_last_rollout_id")

    destroy_process_groups()

    if self.args.rematerialize_param_from_master_weight and self.role == "actor":
        # Params stay resident for update_weights, which pauses them afterwards.
        torch_memory_saver.pause(tag="grad_buffer")
```

```

        torch_memory_saver.pause(tag="default")
    elif lora_rollout_enabled(self.args):
        # adapter params keep their host backup in "default"; the grad buffers have none
        # 检查 grad_buffer 是否已被 offload_grad_buffer() 暂停, 避免重复操作
        if not self._grad_buffer_paused:
            torch_memory_saver.pause(tag="grad_buffer")
            self._grad_buffer_paused = True
            torch_memory_saver.pause(tag="default")
        else:
            torch_memory_saver.pause(tag=None)

    self._asleep = True
    print_memory("after offload model")

    if should_log_cpu_memory:
        log_cpu_memory(self._last_rollout_id, self.args, "after_offload_train")

# 新增的 offload_grad_buffer() 方法, 用于在 gpu 模式下提前释放 LoRA 梯度缓冲区
@with_logs
@timer
def offload_grad_buffer(self) -> None:
    """Free the LoRA grad buffers ahead of sleep(), so the engine can resume its weights first."""
    assert self.args.offload_train
    assert lora_rollout_enabled(self.args), "only LoRA keeps its grad buffers in a no-backup region"
    if self._asleep or self._grad_buffer_paused:
        return
    print_memory("before offload grad buffer")
    torch_memory_saver.pause(tag="grad_buffer")
    self._grad_buffer_paused = True
    print_memory("after offload grad buffer")

```

tests/fast/backends/megatron_utils/test_shared_ppo_lifecycle.py

核心测试文件, 新增了验证 LoRA 梯度缓冲区生命周期管理的单元测试, 确保了实现的正确性。

```

# 新增的测试: 验证 LoRA sleep 正确暂停 grad_buffer 和 default
def test_lora_sleep_pauses_the_grad_buffers_and_wake_up_resumes_them(actor_module,
monkeypatch):
    """The LoRA grad buffers have no host backup, so a sleep that leaves them out keeps them on the GPU."""
    worker, saver, _ = _lifecycle_worker(actor_module, monkeypatch, asleep=False)
    monkeypatch.setattr(actor_module, "lora_rollout_enabled", lambda _args: True)

    worker.sleep()
    worker.wake_up()

    # 验证暂停顺序: 先 grad_buffer, 后 default
    assert saver.pause.call_args_list == [call(tag="grad_buffer"), call(tag="default")]
    # 验证恢复顺序: 先 default, 后 grad_buffer

```

```

assert saver.resume.call_args_list == [call(tag="default"), call(tag="grad_buffer")]

# 新增的测试：验证 offload_grad_buffer 幂等性
def test_offload_grad_buffer_frees_them_once_and_sleep_does_not_pause_them_again(actor_
module, monkeypatch):
    """offload_grad_buffer() and sleep() must pause the grad buffers exactly once between them.
    """

    worker, saver, _ = _lifecycle_worker(actor_module, monkeypatch, asleep=False)
    monkeypatch.setattr(actor_module, "lora_rollout_enabled", lambda _args: True)

    # 多次调用 offload_grad_buffer，验证只暂停一次
    worker.offload_grad_buffer()
    worker.offload_grad_buffer()
    worker.sleep()
    worker.wake_up()

    # 验证 grad_buffer 只被暂停一次，且 sleep() 不会重复暂停
    assert saver.pause.call_args_list == [call(tag="grad_buffer"), call(tag="default")]
    assert saver.resume.call_args_list == [call(tag="default"), call(tag="grad_buffer")]
    assert worker._grad_buffer_paused is False

```

评论区精华

评论中核心讨论点是 LoRA 修改对 **multi-lora** 路径的潜在影响。**xyuzh** 提问：'Would this impact multi-lora? grad_buffer pause with no-backup could unmap DDP main_grad storage'。作者 **yueming-yuan** 明确回复：'It would not, this does not touch the multi-lora path'。这确认了变更的隔离性，消除了对现有功能的担忧。评审已通过，未发现其他设计争议。

- LoRA 修改对 multi-lora 路径的影响 (correctness): 确认变更仅影响 LoRA rollout 路径，不影响 multi-lora。已解决。

风险与影响

- 风险：
 1. 内存管理正确性风险：sleep 和 wake_up 方法中对 "grad_buffer" 和 "default" 标签的暂停 / 恢复顺序和幂等性至关重要。错误的顺序或重复操作可能导致内存泄漏或访问已释放内存。测试覆盖了主要序列，但对异常路径（如 offload_grad_buffer 后的崩溃恢复）覆盖有限。
 2. 状态同步风险：_grad_buffer_paused 状态标志需要在分布式 actor 间保持一致，但当前实现依赖于每个 actor 的本地状态。在存在 actor 故障恢复的场景下，该状态可能不一致。
 3. 兼容性风险：虽然 multi-lora 路径未受影响，但任何依赖 torch_memory_saver 暂停逻辑的模块（如自定义优化器或梯度累积）都可能受到新行为影响，需要关注。- 影响：本 PR 影响使用 --colocate-memory-peak-device gpu 和 LoRA 的训练场景，特别是对于内存受限的大模型训练（如 GB300 K3），能显著降低训练 / 推理切换时的内存峰值。它

解锁了 LoRA 在 GPU 内存共享模式下的应用，扩展了系统的部署灵活性。变更集中在内存管理核心路径，但通过明确的 `lora_rollout_enabled` 守卫，对非 LoRA 和全参数训练的逻辑流无影响。 - 风险标记：核心路径变更，内存管理状态同步，测试覆盖有限

关联脉络

- PR #2781 `--colocate-memory-peak-device gpu`: 本 PR 的基础功能，实现了 gpu 模式但初始不支持 LoRA。本 PR 为其添加了 LoRA 支持。
- PR #1922 Kimi K3 work (implied): 本 PR 从 Kimi K3 工作中拆分出来，服务于 K3 在 GB300 上的全模型训练需求。
- PR #1825 Implied foundational PR: 本 PR 的另一个堆叠基础。