

PR #3089 完整报告

radixark/miles

fix colocate lora duplicated cpu param backup by keeping TMS only

合并时间: 2026-09-03 07:18

原文链接: <http://prhub.com.cn/radixark/miles/pull/3089>

执行摘要

- 一句话: 优化 colocate LoRA 权重同步, 消除重复的 CPU 参数备份, 节省大量主机内存。
- 推荐动作: 本 PR 是一个针对特定高性能计算场景 (colocate LoRA) 的重要内存优化。实现逻辑清晰, 通过引入条件判断消除了已知的内存冗余, 并提供了详实的性能对比数据。建议所有使用 colocate LoRA 进行训练的团队关注并考虑集成。代码变更主要集中在两个文件, 且通过提交历史可以看出作者对边界条件 (如 disk offload、CI 兼容性) 进行了细致处理, 值得精读其设计决策。

功能与动机

在 colocate LoRA 训练中, SGLang 引擎需要在 CPU 侧存储一份训练权重的备份副本。`torch_memory_saver` 在 `trainer sleep()` 模式下已经为整个模型创建了一个 CPU 备份。原有逻辑通过 `weights_backuper` 为 `actor` 额外创建了一个 pinned 的 CPU 副本, 造成了冗余存储。PR body 中的性能验证表明, 这一冗余导致在 8xH200 节点上多占用了约 90-100 GB 的主机内存, 是一个需要修复的效率问题。

实现拆解

1. 新增权重同步路径判断: 在 `actor.py` 中, 新增了 `_weight_sync_reads_tms_backup` 计算属性。当满足以下所有条件时返回 `True`: 启用 `--colocate`、启用 LoRA、`--offload-train-target` 为 `cpu`, 且无需其他模型副本 (如 `ref`、`teacher`、`old_actor`)。这用于决定权重同步是否可以直接读取 `torch_memory_saver` 的备份。
2. 条件化权重同步器获取方式: 在 `actor.py` 的 `init` 方法中, 根据 `_weight_sync_reads_tms_backup` 的结果, 为 `weight_updater` 提供不同的 `weights_getter lambda`。若为 `True`, 则通过新的 `named_params_and_buffers` 函数 (带 `translate_gpu_to_cpu=True`) 直接获取备份; 否则沿用原来的 `weights_backuper.get("actor")`。
3. 扩展通用权重枚举接口: 在 `update_weight/common.py` 的 `named_params_and_buffers` 函数中新增 `translate_gpu_to_cpu` 参数。当该参数为 `True` 时, 通过新增的 `_maybe_get_cpu_backup` 辅助函数, 利用 `torch_memory_saver.get_cpu_backup(tensor, zero_copy=True)` 零拷贝获取 tensor 的 CPU 备份; 若无备份则返回原 tensor。
4. 调整权重备份触发条件: 修改 `actor.py` 中的 `_enable_weight_backup` 属性, 使其在 `_weight_sync_reads_tms_backup` 为 `True` 时返回 `False`, 从而在 `reconcile_adapters` 中加载 / 清理适配器时, 跳过不必要的 `weights_backuper.backup("actor")` 调用, 避免创建

冗余副本。

5. 边界情况处理：提交历史显示，对 `--offload-train-target disk` 的情况进行了专门处理，确保在此配置下仍使用 `pinned actor` 副本，因为此时 `torch_memory_saver` 不保留 CPU 备份。同时，为解决 CI 兼容性问题，`torch_memory_saver` 的导入被设为惰性导入。

关键文件：

- `miles/backends/megatron_utils/actor.py`（模块 训练器后端；类别 `source`；类型 `core-logic`；符号 `_weight_sync_reads_tms_backup`, `_enable_weight_backup`）：核心变更文件，新增了判断是否读取 `torch_memory_saver` 备份的逻辑（`_weight_sync_reads_tms_backup`），并据此调整了权重同步器的获取方式和备份触发条件，是优化内存占用的关键。
- `miles/backends/megatron_utils/update_weight/common.py`（模块 权重更新工具；类别 `source`；类型 `core-logic`；符号 `named_params_and_buffers`, `_maybe_get_cpu_backup`）：提供了实现零拷贝读取的基础工具函数。新增 `translate_gpu_to_cpu` 参数和 `_maybe_get_cpu_backup` 辅助函数，是 `actor.py` 新逻辑得以实现的基础。

关键符号：`_weight_sync_reads_tms_backup`, `_enable_weight_backup`, `named_params_and_buffers`, `_maybe_get_cpu_backup`

关键源码片段

`miles/backends/megatron_utils/actor.py`

核心变更文件，新增了判断是否读取 `torch_memory_saver` 备份的逻辑（`_weight_sync_reads_tms_backup`），并据此调整了权重同步器的获取方式和备份触发条件，是优化内存占用的关键。

```
# actor.py
@property
def _weight_sync_reads_tms_backup(self) -> bool:
    """Under colocated LoRA the frozen base already has a memory-saver host backup; a
    pinned "actor" copy of it would duplicate the whole base per rank. Model switching
    still needs the real backups, and a disk offload target leaves no host backup to read."""
    return (
        self.args.colocate
        and is_lora_enabled(self.args)
        and self.args.offload_train_target == "cpu"
        and not (self.with_ref or self.with_opd_teacher or self.args.keep_old_actor)
    )

@property
def _enable_weight_backup(self) -> bool:
    """Weight backup is only needed for CPU-side model switching or colocated tensor weight
    sync."""
    # When weight sync reads directly from the torch_memory_saver backup,
    # an additional pinned actor copy is not needed.
    if self._weight_sync_reads_tms_backup:
        return False
```

```
return self.with_ref or self.with_opd_teacher or self.args.keep_old_actor or self.args.colocate
```

在 init 方法中, weight_updater 的初始化逻辑如下:

```
if self._weight_sync_reads_tms_backup:
```

```
    # 零拷贝读取 torch_memory_saver 的 CPU 备份, 避免创建新的 pinned 副本
```

```
    weights_getter = lambda: dict(
        named_params_and_buffers(
            self.args,
            self.model,
            convert_to_global_name=args.megatron_to_hf_mode == "raw",
            translate_gpu_to_cpu=True,
        )
    )
```

```
else:
```

```
    weights_getter = lambda: self.weights_backuper.get("actor")
```

miles/backends/megatron_utils/update_weight/common.py

提供了实现零拷贝读取的基础工具函数。新增 `translate_gpu_to_cpu` 参数和 `_maybe_get_cpu_backup` 辅助函数, 是 `actor.py` 新逻辑得以实现的基础。

```
# update_weight/common.py
```

```
def named_params_and_buffers(
```

```
    args: Namespace,
```

```
    model: Sequence[torch.nn.Module],
```

```
    convert_to_global_name: bool = True,
```

```
    # 新增参数: 控制是否将 GPU tensor 转换为其 torch_memory_saver 的 CPU 备份
```

```
    translate_gpu_to_cpu: bool = False,
```

```
) -> Iterator[tuple[str, torch.Tensor]]:
```

```
    if convert_to_global_name:
```

```
        ans = _named_params_and_buffers_global(args, model)
```

```
    else:
```

```
        ans = _named_params_and_buffers_vanilla(model)
```

```
    # 如果启用转换, 对每个 tensor 应用零拷贝读取 CPU 备份的逻辑
```

```
    if translate_gpu_to_cpu:
```

```
        ans = ((name, _maybe_get_cpu_backup(tensor)) for name, tensor in ans)
```

```
    return ans
```

```
def _maybe_get_cpu_backup(x: torch.Tensor):
```

```
    """尝试获取 torch_memory_saver 中的 CPU 备份 (零拷贝), 若不存在则返回原 tensor。"""
```

```
    from torch_memory_saver import torch_memory_saver
```

```
    # zero_copy=True 意味着不复制数据, 直接返回备份的引用 (如果存在)
```

```
    if (cpu_tensor := torch_memory_saver.get_cpu_backup(x, zero_copy=True)) is not None:
```

```
        return cpu_tensor
```

return x

评论区精华

无活跃的 review 讨论。核心实现细节和权衡体现在提交信息中，特别是初始提交和后续几个修正提交。作者通过一系列迭代解决了 CI 构建问题（惰性导入）、明确了 `disk offload` 目标的例外情况，并最终确立了清晰的命名（`translate_gpu_to_cpu`）。

- 暂无高价值评论线程

风险与影响

- 风险：
 1. 条件逻辑复杂性：`_weight_sync_reads_tms_backup` 属性依赖于四个参数的组合判断。如果后续配置选项增加或现有语义变化，此条件可能需要同步维护，否则可能导致错误的权重同步路径选择。
 2. 零拷贝依赖的正确性：依赖 `torch_memory_saver.get_cpu_backup(x, zero_copy=True)` 返回正确的、未过时的 CPU 张量。如果 `torch_memory_saver` 内部状态管理出错（如备份被意外释放），可能导致 `update_weights` 读取到无效数据。
 3. 多 LoRA 适配器场景影响：在 `reconcile_adapters` 中，备份操作现在被 `_enable_weight_backup` 条件守护。虽然当前逻辑（`_weight_sync_reads_tms_backup` 为 `True` 时 `_enable_weight_backup` 为 `False`）能确保不备份，但未来如果权重备份的依赖条件发生变化，可能需要重新审视此处逻辑。
 - 影响：用户影响：直接影响使用 `--colocate`、LoRA 和 `--offload-train-target cpu` 进行训练的用户。该场景下的训练任务将获得显著的主机内存占用下降（约 10-12 GB/rank），可能允许在相同硬件上运行更大模型或增加批处理大小。系统影响：修改了训练器（trainer）在权重同步和模型切换时的核心内存管理逻辑，属于训练核心路径的性能优化。对推理引擎（SGLang）侧无影响（如验证数据显示 `sglang scheduler RSS is unchanged`）。团队影响：为 `colocate` LoRA 训练路径带来了重要的资源效率提升。变更集中且逻辑自洽，降低了长期维护的内存开销成本。
- 风险标记：核心路径变更，条件逻辑复杂性，依赖外部库状态，缺少直接测试覆盖

关联脉络

- PR #2223 部分恢复自该 PR: 当前 PR body 明确说明 'Partially recovered from <https://github.com/radixark/miles/pull/2223>', 表明这是对历史 PR（可能涉及类似优化但未完全合并或已回退）的延续和重新实现。
- PR #2781 Add `--colocate-memory-peak-device: overlap the trainer/rollout handoff on the GPU`: 同属于 `colocate` 模式下的内存优化主线。PR#2781 通过调整交接顺序来缓解内存峰值，而本 PR 则通过消除冗余备份来直接降低内存占用基线，二者目标一致，都是优化训练器（trainer）在 `colocate` 模式下的资源使用。