

PR #3141 完整报告

radixark/miles

Dump debug train data once per DP/CP shard instead of on every rank

合并时间: 2026-09-08 14:10

原文链接: <http://prhub.com.cn/radixark/miles/pull/3141>

执行摘要

- 一句话: 优化训练调试数据写入策略, 仅每个 DP/CP 分片写一次, 大幅减少磁盘占用。
- 推荐动作: 这是一个高质量、目标明确的优化和修复 PR。它不仅解决了实际痛点 (磁盘空间浪费), 还确保了功能的向后兼容性和测试覆盖。对于处理分布式训练调试数据的工程师来说, 其门控写入和可见性探测的修复逻辑非常有参考价值。建议精读 `save_debug_train_data` 和 `DumpReader._visible` 的变更, 理解其并行状态判断的设计。

功能与动机

在使用 `--dump-details` 参数进行长周期训练时, 所有训练 rank 都会将各自的 rollout 分片写入磁盘。然而, 由于 Tensor Parallelism (TP) 的对等 rank 持有相同数据, 且 Pipeline Parallelism (PP) 中仅最后一个 stage 计算 Dashboard 需要的 per-token 字段 (如 `log_probs`, `entropy`), 导致磁盘上产生了 `tp_size * pp_size` 份冗余副本。Dashboard 读者在加载时需要先进行去重, 这浪费了磁盘空间并可能导致磁盘写满。

实现拆解

1. 核心训练数据写入逻辑改造: 在 `miles/utils/train_dump_utils.py` 的 `save_debug_train_data` 函数中, 引入并行状态检查。仅在 TP rank 为 0 且处于最后一个 PP stage 时才执行写入。这样确保每个 DP/CP 分片只写入一次。
2. 策略损失调试输出优化: 在 `miles/backends/training_utils/debug_dump.py` 的 `maybe_dump_policy_loss_debug` 函数中, 同样增加了对 `tp.rank == 0` 的检查, 跳过 TP 对等 rank 的重复计算和文件写入。
3. Dashboard 可见性探测修复: 修改 `miles/dashboard/dump_reader.py` 的 `DumpReader._visible` 方法, 将检查特定 rank 0 文件的逻辑改为 glob 搜索任意 rank 文件, 以适应 PP > 1 时 global rank 0 不再是写入者的场景。
4. 测试配套更新与新增: 新增 `tests/fast/utils/test_train_dump_utils.py` 文件, 参数化测试验证写入逻辑。更新 `tests/fast/dashboard/test_dump_reader_join.py` 中的测试, 使用非 rank-0 文件来验证可见性逻辑的健壮性。

关键文件:

- `miles/utils/train_dump_utils.py` (模块 训练数据转储; 类别 source; 类型 core-logic; 符号 `save_debug_train_data`): 核心训练调试数据写入逻辑的改造点, 实现了减少冗余写入的核心功能。

- miles/dashboard/dump_reader.py (模块 数据读取; 类别 source; 类型 core-logic; 符号 DumpReader._visible) : 修复了 Dashboard 读者在 $pp > 1$ 场景下的可见性探测逻辑, 是确保新写入策略下功能正确性的关键。
- tests/fast/utils/test_train_dump_utils.py (模块 测试; 类别 test; 类型 test-coverage; 符号 _parallel_state, test_only_one_rank_per_shard_writes) : 新增的单元测试, 直接验证了核心变更逻辑的正确性, 覆盖了不同并行状态下的写入行为。

关键符号: save_debug_train_data, maybe_dump_policy_loss_debug, DumpReader._visible

关键源码片段

miles/utils/train_dump_utils.py

核心训练调试数据写入逻辑的改造点, 实现了减少冗余写入的核心功能。

```
def save_debug_train_data(args, *, rollout_id, rollout_data):
    if args.save_debug_train_data is not None:
        parallel_state = get_parallel_state()
        # 仅当 TP rank 为 0 且处于最后一个 PP 阶段时才写入, 避免冗余文件。
        # TP 对等 rank 持有相同数据, 且只有最后一个 PP 阶段计算 per-token 字段。
        if parallel_state.tp.rank != 0 or not parallel_state.is_pp_last_stage:
            return
        save_debug_train_data_for_rank(
            args,
            rollout_id=rollout_id,
            rollout_data=rollout_data,
            rank=torch.distributed.get_rank(),
            cp_rank=parallel_state.cp.rank,
            cp_size=parallel_state.cp.size,
        )
```

评论区精华

该 PR 仅由 Claude Bot 进行了自动评论, 并由 Zhichenzzz 批准, 没有详细的技术讨论。变更逻辑清晰, 主要集中在对并行状态判断的应用。

- 并行状态门控写入的实现 (design): 在 save_debug_train_data 和 maybe_dump_policy_loss_debug 中增加 `tp.rank == 0` 检查, 在 save_debug_train_data 中增加 `is_pp_last_stage` 检查。
- Dashboard 可见性探测逻辑的修复 (correctness): 将硬编码的文件检查改为使用 glob 搜索任意匹配文件, 确保在 $pp > 1$ 时也能正确探测到新数据。

风险与影响

- 风险:
 1. 兼容性风险: 旧格式的全 rank dump 文件仍可被正确读取, 但新格式的 dump 文件不再包含冗余副本。对于依赖旧 dump 文件特定文件名 (如 rollout_id_0.pt) 的外部工具,

可能需要适配。

2. 逻辑正确性风险：条件判断 `tp.rank != 0 or not is_pp_last_stage` 是实现的关键。需要确保 `parallel_state` 在所有训练场景下都已正确初始化。FSDP 场景 (`tp.size == 1`) 不受影响，逻辑与之前一致。
3. Dashboard 行为变更：`_visible` 方法从检查固定文件变为 `glob` 搜索，这增加了 I/O 操作，但影响范围仅限于判断新步骤是否可见的短暂窗口，性能影响可忽略。
 - 影响：用户影响：显著减少了使用 `--dump-details` 时的磁盘空间占用，提升了长周期训练的稳定性。
 - 对于开发者，调试流程更加高效，不再需要处理大量冗余文件。
 - 系统影响：改善了训练集群的磁盘利用率和 I/O 性能。对 Dashboard 的数据加载和展示逻辑无负面影响，且通过修复可见性探测提升了可靠性。
 - 团队影响：为并行训练下的调试数据管理提供了更优的实践，相关模式（基于并行状态进行门控写入）可被后续功能参考。

- 风险标记：核心路径变更，数据格式兼容性

关联脉络

- PR #3114 Tolerate missing optional SGLang Anthropic helpers: 同属 bugfix 类型，关注系统的容错和健壮性，但功能模块不同（路由 vs 训练数据）。
- PR #3091 feat(megatron): reset optimizer state by optimizer class: 同属对核心训练流程（优化器状态）的优化和修复，涉及配置和测试，体现了对训练稳定性的持续改进。