PR #1926 完整报告

THUDM/slime

Move micro-batch scheduling from training side to rollout side

合并时间：2026-05-20 20:52

原文链接：http://prhub.com.cn/THUDM/slime/pull/1926

执行摘要

- 一句话：将微批次调度从训练侧移至 rollout 侧，全局优化 bin packing。
- 推荐动作：值得精读，特别是 slime/utils/dp_schedule.py 的设计思路和 tests/test_dp_schedule.py 中不变量断言。需要关注：
  - 如何将纯逻辑从框架依赖中拆解出来（纯函数 + CPU 测试）。
  - expand_bins_by_splitting 如何保持 bin 容量约束。
  - 统一 static 和 dynamic 路径的 micro_batch_indices 数据契约。

功能与动机

之前 get_data_iterator 在 actor 侧逐 DP rank 计算每步的微批次调度，然后通过 all-reduce 以 MAX 同步 num_microbatches。在样本长度不均衡时会导致某些 rank 分配了超过实际所需的 mbs，且 per-rank 本地均衡比全局视图掌握的信息更少。

实现拆解

实现共分 5 步：

1. 新建 slime/utils/dp_schedule.py 模块 将纯调度逻辑（build_dp_schedule 和 compute_dynamic_global_batch_size）从 Ray/SGLang 依赖中解耦，使其可在纯 CPU 环境下进行单元测试。
2. 扩展 slime/utils/seqlen_balancing.py 添加 first_fit_pack、_split_bin_by_tokens、expand_bins_by_splitting 函数，支持全局 first-fit bin packing 以及按 token 数从最大 bin 分裂扩增，从而避免创建超出容量上限的新 bin。
3. 重构 _split_train_data_by_dp（slime/ray/rollout.py） 调用 compute_dynamic_global_batch_size 确定有效 global batch size，trim 样本，然后调用 build_dp_schedule 获得 partitions 和 micro_batch_indices。不再依赖 _compute_dynamic_global_batch_size 和 all-reduce。
4. 简化 DataIterator（slime/backends/megatron_utils/data.py） 移除 micro_batch_size 分支，只保留 micro_batch_indices 驱动的方式，get_data_iterator 直接读取 rollout_data 中的 schedule。同时删除不再需要的 get_minimum_num_micro_batch_size（slime/utils/data.py）。
5. 编写 CPU 单元测试并更新 CI 新增 tests/test_dp_schedule.py，覆盖 static/dynamic/VPP/oversize/balance 等场景，验证常量化不变量。在 .

github/workflows/pr-test.yml.j2 中添加 cpu-unittest job。

关键文件：

- slime/utils/dp_schedule.py（模块 DP 调度；类别 source；类型 dependency-wiring；符号 compute_dynamic_global_batch_size, build_dp_schedule）：新文件，包含核心调度函数 build_dp_schedule 和 compute_dynamic_global_batch_size，解耦了调度逻辑与 Ray 依赖，是 PR 的核心模块。
- slime/backends/megatron_utils/data.py（模块 数据迭代器；类别 source；类型 core-logic；符号 get_data_iterator, _generate_data_iterator）：简化 DataIterator，移除 micro_batch_size 分支，统一只接受 micro_batch_indices；get_data_iterator 直接读取 rollout_data 中的调度信息。
- slime/ray/rollout.py（模块 数据收集；类别 source；类型 dependency-wiring；符号 _compute_dynamic_global_batch_size, _split_train_data_by_dp）：重构 _split_train_data_by_dp 以调用 dp_schedule 生成调度，移除 _compute_dynamic_global_batch_size 和 all-reduce 逻辑。
- tests/test_dp_schedule.py（模块 测试；类别 test；类型 test-coverage；符号 make_args, make_tp, assert_invariants, test_static_stride_single_step）：新增 CPU 单元测试，覆盖 build_dp_schedule 的所有不变量条件，确保调度正确性。
- slime/utils/seqlen_balancing.py（模块 序列平衡；类别 source；类型 core-logic；符号 first_fit_pack, _split_bin_by_tokens, expand_bins_by_splitting）：新增 first_fit_pack、_split_bin_by_tokens、expand_bins_by_splitting 函数，为动态调度提供 bin packing 工具。

关键符号：compute_dynamic_global_batch_size, build_dp_schedule, first_fit_pack, _split_bin_by_tokens, expand_bins_by_splitting, get_data_iterator, _split_train_data_by_dp

关键源码片段

tests/test_dp_schedule.py

新增 CPU 单元测试，覆盖 build_dp_schedule 的所有不变量条件，确保调度正确性。

```
def assert_invariants(
    partitions, micro_batch_indices, num_microbatches, *,
    dp_size, total_lengths, max_per_bin=None
):
    """验证 dp_schedule.py 文档中申明的不变量。"""
    expected_per_rank = len(total_lengths) // dp_size
    seen_global: set[int] = set()
    for r in range(dp_size):
        partition = partitions[r]
        mbi = micro_batch_indices[r]

        # 每个 rank 样本数相同
        assert len(partition) == expected_per_rank, (
            f"rank {r}: {len(partition)} samples, want {expected_per_rank}"
```

```
        )

        # 每个 rank 的 mbs 总数等于所有步的 num_microbatches 之和（PP 同步要求）
        assert len(mbi) == sum(num_microbatches), f"rank {r}: mbs count mismatch"

        # 展平 micro_batch_indices 应恰好覆盖 [0, len(partition))
        flat = [i for mbs in mbi for i in mbs]
        assert flat == list(range(len(partition))), (
            f"rank {r}: micro_batch_indices don't tile [0, n)"
        )

        # 各 rank 的 partition 之间无交集且并集覆盖所有样本
        assert seen_global.isdisjoint(partition), f"rank {r}: overlap with other ranks"
        seen_global.update(partition)
    assert seen_global == set(range(len(total_lengths))), "some samples not assigned to any
    rank"

    if max_per_bin is None:
        return

    # 每个 mbs 的 token 数 <= max_per_bin，除非是单个超大样本独占的 bin
    for r in range(dp_size):
        partition = partitions[r]
        for mbs in micro_batch_indices[r]:
            bin_total = sum(total_lengths[partition[i]] for i in mbs)
            if bin_total > max_per_bin:
                assert len(mbs) == 1, (
                    f"rank {r}: mbs sum {bin_total} > {max_per_bin} but contains {len(mbs)} samples"
                )
```

评论区精华

无审核评论。但从 14 次 commit 可以看出设计迭代过程：最初将调度整体移入 rollout → 逐步拆分为 schedule-only 函数 → 将辅助函数移出 ray-tainted 模块以通过 CPU 单测 → 最终统一 static/dynamic 接口。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 核心调度逻辑变更：build_dp_schedule 的全局 bin packing 与原先的 per-rank 调度策略存在差异，可能导致训练步数、mbs 数量发生变化，需要验证不变量（测试已覆盖）。
  2. 接口不兼容：DataIterator.__init__ 移除了 micro_batch_size 参数，任何外部直接构造 DataIterator 的代码都会报错（需确认无外部依赖）。
  3. 配置 key 变更：train_parallel_config 现在要求包含 cp_size、vpp_size、microbatch_group_size_per_vp_stage，旧配置缺少这些字段会导致 KeyError。

4. LR scheduler 步进：dynamic batch 场景下 samples-per-step 可变，需确保 opt_param_scheduler.step 使用实际 processed 样本数（commit 91a57a 已修复）。
5. 缺少集成测试：仅有 CPU 单元测试，无完整 rollout→training 的端到端测试。 - 影响：用户影响：正常训练流程应无感知，但若手动构造 DataIterator 或依赖旧配置 key 的项目需要适配。系统影响：移除了一道 all-reduce 同步点，降低通信开销；动态 batch 下各 rank 的 mbs 数量更均衡，有利于 pipeline 同步，潜在提升吞吐。团队影响：调度逻辑集中到 dp_schedule.py 便于维护和测试；后续新增调度策略时直接扩展该文件。

- 风险标记：核心调度逻辑重构，DataIterator 接口不兼容，配置 key 新增要求，缺少集成测试，LR 调度步进修正

关联脉络

- 暂无明显关联 PR