PR #1933 完整报告

THUDM/slime

[2/N] Support training with variable global batch size

合并时间：2026-05-25 10:37

原文链接：http://prhub.com.cn/THUDM/slime/pull/1933

执行摘要

- 一句话：支持动态全局 batch 大小，重构调度与报告聚合
- 推荐动作：该 PR 是支持动态 batch size 的关键环节，设计巧妙（pack-first-distribute-second），测试完备（涵盖报告不变性和梯度 CP 不变性）。推荐架构师和核心开发精读 cp_utils.py 中的指标聚合设计，以及 dp_schedule.py 中的调度策略。测试代码也值得作为集成测试范本。

功能与动机

当每个 rollout 可能产生不同数量的训练样本（如 compact/subagent 场景）时，原有的固定全局 batch size 机制不再适用，需要每个训练步处理不等量样本，同时保持流水线并行同步和指标正确。本 PR 是系列 [1/N]（#1930）的延续，完善调度和报告逻辑，使动态 batch size 真正可工作。

实现拆解

1. 调度重构（slime/utils/dp_schedule.py）：将调度哲学从“多步均匀分配”改为“先打包后分配”（pack-first-distribute-second）。新增 _pack_step_into_mbs 函数统一动态 / 静态打包逻辑；build_dp_schedule 增加 rollout_indices 参数，先按 rollout 分组后打包再分发到 DP rank，确保同一 rollout 的样本不跨 step。
2. 报告聚合重构（slime/backends/megatron_utils/cp_utils.py）：为 get_sum_of_sample_mean 添加 sample_denoms 参数，支持传入预计算的 per-rollout 分母（同一 rollout 的所有样本共享一个分母）。新增 reduce_train_step_metrics、rollout_log_metric_contribution、gather_and_reduce_log_dict 三个函数，集中处理训练步和 rollout 侧的指标聚合，使报告计算不依赖 micro-batch 划分。
3. 调用方适配：修改 slime/ray/rollout.py、slime/backends/megatron_utils/data.py、model.py 等文件，传递新参数（如 rollout_indices、sample_denoms），并调用新的报告函数代替内联逻辑。同时引入 step_split_hub 模块用于自定义 step 拆分。
4. 测试配套：新增 5 个测试文件（test_metric_report.py、test_metric_report_dist.py、test_loss_cp_invariance.py、test_cp_utils.py、_cp_dist_helpers.py）并修改 test_dp_schedule.py，覆盖单进程和分布式场景下的报告不变性、梯度 CP 不变性、调度正确性。测试使用 CPU 可运行的 Megatron 桩，确保 CI 可执行。

关键文件：

- slime/backends/megatron_utils/cp_utils.py（模块 指标报告；类别 source；类型 core-logic；符号 reduce_train_step_metrics, rollout_log_metric_contribution, gather_and_reduce_log_dict）：核心源码变更：新增 sample_denoms 参数支持 per-rollout 分母，新增 reduce_train_step_metrics 等函数统合指标聚合逻辑。所有训练报告的正确性依赖此处。
- slime/utils/dp_schedule.py（模块 调度器；类别 source；类型 dependency-wiring；符号 compute_dynamic_global_batch_size, _pack_step_into_mbs）：调度核心重构：引入 pack-first-distribute-second 策略，新增 _pack_step_into_mbs 函数；build_dp_schedule 增加 rollout_indices 参数以按 rollout 分组。
- tests/test_metric_report.py（模块 指标测试；类别 test；类型 test-coverage；符号 mock_dp_with_cp_group, _simulate_report, test_per_rollout_mean_report_invariant_to_mb_distribution, test_per_token_loss_report_invariant_to_mb_distribution）：单进程报告不变性测试，验证不同 mb 分布下报告值一致，覆盖 per-rollout-mean 和 per-token-loss 路径。
- tests/test_metric_report_dist.py（模块 分布式指标测试；类别 test；类型 test-coverage；符号 _train_step_distributed_worker, test_train_step_per_rollout_mean_real_distributed, test_train_step_per_token_loss_real_distributed, _rollout_log_distributed_worker）：多进程分布式测试，使用真实 torch.distributed 验证跨 rank 的报告聚合正确性。
- tests/test_loss_cp_invariance.py（模块 CP 不变性测试；类别 test；类型 test-coverage；符号 with, _grad_norm_worker, _run_grad_norm_worker, test_backward_grad_is_cp_invariant）：梯度 CP 不变性测试，验证经过 slime 的 loss prescaling + Megatron 的 per-mb scaling + DDP 的 grad averaging 后，梯度范数与 CP size 无关。

关键符号：get_sum_of_sample_mean, reduce_train_step_metrics, rollout_log_metric_contribution, gather_and_reduce_log_dict, build_dp_schedule, _pack_step_into_mbs, compute_dynamic_global_batch_size

关键源码片段

slime/backends/megatron_utils/cp_utils.py

核心源码变更：新增 sample_denoms 参数支持 per-rollout 分母，新增 reduce_train_step_metrics 等函数统合指标聚合逻辑。所有训练报告的正确性依赖此处。

```
# slime/backends/megatron_utils/cp_utils.py

def get_sum_of_sample_mean(
    total_lengths: list[int],
    response_lengths: list[int],
    loss_masks: list[torch.Tensor],
    sample_denoms: list[torch.Tensor] | torch.Tensor | None = None, # 新增：预计算的 per-sample 分母
    calculate_per_token_loss: bool = False,
    qkv_format: str = 'thd',
    max_seq_lens: list[int] | None = None,
```

```
) -> Callable[[torch.Tensor], torch.Tensor]:
    # Calculate correct sample mean for CP.
    # The default (sample_denoms=None) is the legacy per-sample mean.
    if sample_denoms is None:
        # 默认：每个样本用自己的 mask 和作为分母
        sample_denoms = [m.sum() for m in loss_masks]
    # ... ( 后续根据 CP size 分支处理 )
    return sum_of_sample_mean if not calculate_per_token_loss else sum_of_token
```

slime/utils/dp_schedule.py

调度核心重构：引入 pack-first-distribute-second 策略，新增 _pack_step_into_mbs 函数；build_dp_schedule 增加 rollout_indices 参数以按 rollout 分组。

```
# slime/utils/dp_schedule.py

def _pack_step_into_mbs(
    step_lengths: list[int],
    *,
    use_dynamic_batch_size: bool,
    max_per_bin: int | None,
    micro_batch_size: int | None,
) -> list[list[int]]:
    # Group a step's samples into mbs.
    if use_dynamic_batch_size:
        # 动态 batch：使用 first-fit 打包，每个 bin 不超过 max_per_bin tokens
        assert max_per_bin is not None
        return first_fit_pack(step_lengths, max_per_bin)
    # 静态 batch：固定 micro_batch_size 切分
    assert micro_batch_size is not None
    n = len(step_lengths)
    return [list(range(i, min(i + micro_batch_size, n))) for i in range(0, n, micro_batch_size)]
```

评论区精华

该 PR 无 Review 评论，作者独立开发完成。从 commit 历史看（共 17 次提交，包含多次 bugfix 和 refactor），开发过程中经历了充分的自我迭代和测试调整。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 报告正确性：新的指标聚合逻辑改变了分母计算方式，但大量测试（单进程、分布式、梯度不变性）覆盖了多种 partition 配置，风险较低。
  2. 调度兼容性：build_dp_schedule 接口变更（增加 rollout_indices）要求所有调用方更新；原静态 batch 路径也需验证。配套测试覆盖了动态和静态路径。
  3. 性能影响：动态 first-fit 打包可能增加 CPU 开销，但 bin packing 有望提升 GPU 利用率。需实际基准测试。

4. 数据契约：sample_denoms 需在 actor 侧预先计算并传递，若计算错误会影响报告，但测试中包含了 per-rollout 分母的验证。 - 影响：用户：使用动态全局 batch size 的训练任务将受益于更高效的样本利用（无需填充）。现有静态 batch 用户接口不变（向后兼容），但若使用旧的报告函数需迁移到新接口。系统：调度和报告模块的核心逻辑变更，影响训练循环的各个部分。新增多个测试覆盖关键路径，提升代码健壮性。团队：需要了解新的调度哲学和报告 API，但文档和测试提供了良好参考。

- 风险标记：核心路径变更，接口不兼容，测试依赖 CPU 桩

关联脉络

- PR #1930 [1/N] Support training with variable global batch size: 本 PR 是该系列的第二部分，建立在前序接口和框架之上。PR #1930 引入了基本的数据管道和测试，本 PR 完善调度和报告。
- PR #1926 Move micro-batch scheduling from training side to rollout side: 之前的工作将微批次调度从训练侧移至 rollout 侧，本 PR 在此基础上进一步优化调度策略。
- PR #1897 Migrate internal feature: 该 PR 也涉及训练进度条和 on-policy 优化，与报告系统相关。