

PR #1927 完整报告

radixark/miles

[feat] Support training with variable global batch size

合并时间: 2026-08-05 11:49

原文链接: <http://prhub.com.cn/radixark/miles/pull/1927>

执行摘要

- 一句话: 支持可变 global batch size 训练, 调度迁至 rollout 侧
- 推荐动作: 值得精读。这是 miles 训练数据管线的架构级变更: 调度、计账、指标三条链路同时重构且保持严格 no-op。重点读 build_dp_schedule 的「先打包后分发」两阶段设计、rollout_mask_sums 分母广播的数学, 以及 can_schedule_on_rollout_side 的资格判定列表 (它决定了未来新 backend 的接入方式)。建议配合 THUDM/slime 系列 PR 一起看设计演进, 并关注后续 #2135/#2197 的清理与修复。

功能与动机

PR body 明确说明动机: RL rollout 生产天然就是变长的——dynamic-sampling 过滤会丢样本、agentic/ 多轮 rollout 每次执行产生的训练样本数不定、采集到的批次几乎不会恰好是 global_batch_size 的整数倍。legacy 管线假设固定 global_batch_size (按样本计数)、每个 DP rank 持有相同样本数、per-sample loss 加权, 后果是: 拖尾样本被裁剪 (浪费生成)、一次产出 N 个样本的 rollout 在 loss 里被计 N 次 (长执行被过加权)、兄弟样本可能被拆到不同 step 或裁剪边界。设计跟随 THUDM/slime 的 variable-global-batch-size 系列 (slime #1930/#1933), FLOPs 均衡打包移植 slime #2017/#2029, FFD 打包与 partial final step 是 miles 自己的扩展。

实现拆解

实现分 5 步拆解:

1. 调度侧迁移 (commit 1): 新增 miles/utils/dp_schedule.py 纯 Python 调度模块, build_dp_schedule 一次返回 partitions / micro_batch_indices / num_microbatches / num_rollouts; has_full_schedule_config 检查训练后端是否广告完整并行配置 (dp/cp/vpp/mb_group)。train_data_conversion.py 的 split_train_data_by_dp 签名从 dp_size 改为 train_parallel_config, 通过 can_schedule_on_rollout_side 分流: 可调度则走 split_train_data_by_dp_scheduled_raw 并把布局写入 shard; 否则保留 legacy split_train_data_by_dp_raw (multi-LoRA、fsdp/torchdynamo、indep_dp/FT、多模态)。训练侧 get_data_iterator 改为消费预计算布局, 删掉原来跨 rank 的 all_reduce(MAX) 同步点。
2. 以 rollout 为单位的 step 规划与打包: 样本按 Sample.rollout_id (缺省回落 sample.index) 聚合, global_batch_size 按 rollout 计数, 同一 rollout 的兄弟样本永不跨 step。动态路径下用新增 first_fit_decreasing_pack 全局 FFD 打包到

`max_tokens_per_gpu * cp` 预算，再通过 `expand_bins_by_splitting` 把 micro-batch 数对齐到 `dp_size * mb_group` (VPP 场景需要)；随后按 `strided`、Karmarkar-Karp (`--balance-data`) 或 FLOPs 均衡 (`--balance-by-flops`) 分发到 DP rank。静态路径按固定 `micro_batch_size` 切块，若无法对齐则启动即失败 (commit 5 / #2197)。

3. 每 rollout 的 loss 记账：`convert_samples_to_train_data` 新增 `rollout_ids` 与 `rollout_mask_sums` (`_compute_rollout_mask_sums` 把整次 rollout 的 mask 总数广播给每个兄弟样本)；`cp_utils.get_sum_of_sample_mean` 新增 `denominators` 参数，`loss.py` 用 rollout 分母替换 per-sample 分母，使散落在不同 micro-batch、DP、CP rank 的 `sum/分母` 在既有 all-reduce 下精确重建为每 rollout 一次 token 加权均值；metric 侧 `log_utils` 改为 (sum, count) 元组 + `reduce_gathered_log_dict` 的 count-weighted 归约，`aggregate_train_losses` 新增 `num_rollouts` 分支。
4. 参数与资格校验：`miles/utils/arguments.py` 新增 `--allow-partial-train-step`、`--balance-by-flop` 等参数；`can_schedule_on_rollout_side` 明确排除 multi-LoRA、多模态、配置不全的 backend；`rollout_data_conversion.py` 新增 `validate_compact_rollout_ids` 保证 compact 形状下兄弟样本必须共享 `rollout_id`。
5. 测试与 CI 配套：CPU 侧覆盖调度不变式（覆盖完备、无重叠、PP 同步、超大样本独享 bin、VPP 对齐、尾部裁剪、partial step、rollout 分组）、FFD 性质、denominator 等价性、归约语义；GPUe2e 三个：`test_qwen2.5_0.5B_compact_rollout.py` (compact 拆分)、`test_qwen3_4B_variable_gbs.py` (gbs=24 训成 [24,8])、`test_qwen3_30B_A3B/test_dp_attention.py` (dp-attention 顺序不敏感)。

关键文件：

- `miles/utils/dp_schedule.py` (模块 调度器；类别 source；类型 core-logic；符号 `build_dp_schedule`, `has_full_schedule_config`, `_calculate_workloads`)：新增的纯 CPU 调度器核心：按 rollout 聚合样本、规划每步 rollout 数、FFD 打包、对齐到 `dp*vpp`、分发到各 DP rank，是整个 PR 的地基。
- `miles/ray/rollout/train_data_conversion.py` (模块 数据转换；类别 source；类型 core-logic；符号 `split_train_data_by_dp`, `can_schedule_on_rollout_side`, `split_train_data_by_dp_scheduled_raw`, `_compute_rollout_mask_sums`)：调度资格判定与数据契约扩展的枢纽：新增 `rollout_ids` / `rollout_mask_sums` / `num_microbatches` / `micro_batch_indices` / `num_rollouts` 五个键，并实现 rollout 侧调度与 legacy 路径的分流。
- `miles/utils/seqlen_balancing.py` (模块 打包层；类别 source；类型 core-logic；符号 `first_fit_pack`, `first_fit_decreasing_pack`, `_first_fit`, `expand_bins_by_splitting`)：新增 first-fit / FFD 装箱与基于 LPT 的 bin 拆分扩容，是动态打包与 align 对齐的底层工具，被 `build_dp_schedule` 直接依赖。
- `miles/backends/training_utils/log_utils.py` (模块 指标归约；类别 source；类型 core-logic；符号 `reduce_gathered_log_dict`, `gather_log_data`, `log_rollout_data`, `aggregate_train_losses`)：指标归约语义从 mean-of-rank-means 改为 count-weighted (sum, count)，`aggregate_train_losses` 支持按 `num_rollouts` 归一，是 per-rollout 记账在日志侧的落地。
- `miles/backends/training_utils/cp_utils.py` (模块 并行工具；类别 source；类型 data-contract；符号 `get_sum_of_sample_mean`)：`get_sum_of_sample_mean` 新增

denominators 参数, loss 分母从 per-sample 改为 per-rollout, 是计账修正的数学核心。

- miles/backends/training_utils/loss.py (模块 损失计算; 类别 source; 类型 data-contract) : loss reducer 接入 rollout 分母, 使跨 micro-batch 的部分贡献重建为每 rollout 一次 token 加权均值。
- miles/backends/training_utils/data.py (模块 数据接口; 类别 source; 类型 dependency-wiring; 符号 get_num_rollouts) : 新增 get_num_rollouts, 把真实 per-step rollout 数从 shard 穿到归一化与 LR 调度。
- miles/args/arguments.py (模块 参数解析; 类别 source; 类型 configuration) : 新增 --allow-partial-train-step、--balance-by-flops 等参数开关, 并承载相关参数校验。
- miles/ray/rollout/rollout_data_conversion.py (模块 数据校验; 类别 source; 类型 core-logic; 符号 validate_compact_rollout_ids) : 新增 validate_compact_rollout_ids, 在 flatten 前强制 compact 兄弟样本共享 rollout_id, 保证计账分组正确。
- tests/fast/utils/test_dp_schedule.py (模块 调度测试; 类别 test; 类型 test-coverage; 符号 assert_invariants, test_static_stride_single_step, test_dynamic_with_vpp_rounds_to_mb_group, test_rollout_grouping_keeps_samples_together) : 440 行 CPU 单测覆盖调度不变式与静态 / 动态 / 平衡 / VPP / 裁剪 / 分组场景, 是调度器正确性的主要保证。
- tests/fast/backends/training_utils/test_rollout_mask_sums.py (模块 计账测试; 类别 test; 类型 test-coverage; 符号 TestRolloutMaskSumsPrecompute, TestDenominators, TestAggregateTrainLossesNumRollouts) : 验证 rollout_mask_sums 预计算、denominator 与 legacy per-sample 的严格等价、per-rollout loss 归约, 是计账数学的回归防线。
- tests/e2e/short/test_qwen2.5_0.5B_compact_rollout.py (模块 紧凑回滚; 类别 test; 类型 test-coverage; 符号 prepare, execute) : compact 拆分场景的端到端验证: 每个 execution 拆成两个共享 rollout_id 的训练样本, 验证按 rollout 计 step 与 loss 加权。
- tests/e2e/megatron/test_qwen3_4B_variable_gbs.py (模块 可变批大小; 类别 test; 类型 test-coverage; 符号 prepare, execute) : 可变 gbs 主路径 e2e: gbs=24 采集 32 个 rollouts, 配合 --allow-partial-train-step 与 cp2 训练成 [24, 8] 两个不均匀 step。

关键符号: build_dp_schedule, has_full_schedule_config, _calculate_workloads, can_schedule_on_rollout_side, split_train_data_by_dp, split_train_data_by_dp_scheduled_raw, split_train_data_by_dp_raw, _compute_rollout_mask_sums, _package_shards, first_fit_pack, first_fit_decreasing_pack, _first_fit, expand_bins_by_splitting, _split_bin_by_tokens, reduce_gathered_log_dict, gather_log_data, log_rollout_data, aggregate_train_losses, get_sum_of_sample_mean, get_num_rollouts, validate_compact_rollout_ids

关键源码片段

miles/ray/rollout/train_data_conversion.py

调度资格判定与数据契约扩展的枢纽: 新增 rollout_ids / rollout_mask_sums / num_microbatches / micro_batch_indices / num_rollouts 五个键, 并实现 rollout 侧调度与

legacy 路径的分流。

```
def can_schedule_on_rollout_side(args, data, train_parallel_config):
    """判定是否能在 rollout 侧预计算完整 DP/microbatch 调度。"""
    if not has_full_schedule_config(train_parallel_config):
        return False # fsdp/torchtitan 只暴露 dp_size, 走 legacy
    if is_multi_lora_enabled(args):
        return False # slot 连续性约束, 训练侧调度保留
    if "multimodal_train_inputs" in data:
        return False # 多模态序列长度在训练侧才会展开
    if "rollout_ids" not in data:
        return False
    global_batch_size = data.get("dynamic_global_batch_size", args.global_batch_size)
    return len(set(data["rollout_ids"])) >= global_batch_size # 至少凑满一个 step

def split_train_data_by_dp(args, data, train_parallel_config):
    """按 DP 切分并写入对象存储; 能预计算调度时走 rollout 侧, 否则回退 legacy。"""
    if can_schedule_on_rollout_side(args, data, train_parallel_config):
        shards = split_train_data_by_dp_scheduled_raw(args, data, train_parallel_config=train_parallel_config)
    else:
        shards = split_train_data_by_dp_raw(args, data, dp_size=train_parallel_config["dp_size"])
    store = object_store.get_instance()
    return [store.put(value=shard, value_spec=ROLLOUT_DATA_VALUE_SPEC) for shard in shards]

def split_train_data_by_dp_scheduled_raw(args, data, *, train_parallel_config):
    """在 rollout 侧跑调度器, 把 micro-batch 布局打进每个 rank 的 shard。"""
    total_lengths = [len(t) for t in data["tokens"]]
    data["total_lengths"] = total_lengths
    global_batch_size = data.get("dynamic_global_batch_size", args.global_batch_size)
    partitions, micro_batch_indices, num_microbatches, num_rollouts = build_dp_schedule(
        args, train_parallel_config, total_lengths,
        global_batch_size=global_batch_size,
        rollout_indices=data["rollout_ids"],
    )
    logger.info(f"Rollout-side DP schedule: num_samples={len(total_lengths)}, "
               f"num_rollouts={num_rollouts}, num_microbatches={num_microbatches}")
    shards = _package_shards(args, data, partitions)
    for rank, shard in enumerate(shards):
        shard["num_microbatches"] = num_microbatches
        shard["micro_batch_indices"] = micro_batch_indices[rank]
        shard["num_rollouts"] = num_rollouts
    return shards

def _compute_rollout_mask_sums(rollout_ids, loss_masks):
```

"""每个样本携带其整次 rollout 的 loss mask 总和，作为 loss 分母。

同一 rollout 的兄弟样本即使散落在不同 micro-batch、不同 DP/CP rank，各自贡献的 sum/分母 也能在既有 all-reduce 下重新拼成一次 rollout 均值。

"""

```
totals = {}
for rid, mask in zip(rollout_ids, loss_masks, strict=True):
    totals[rid] = totals.get(rid, 0) + sum(mask)
return [totals[rid] for rid in rollout_ids]
```

评论区精华

review 中最有价值的交锋集中在两点：

- 命名歧义（已解决）：guapisolo 指出 `mbs is confusing "micro batch size" or "micro batches"`，并亲自提交 commit f66b78a 将 `mbs/step_mbs/num_mbs/rank_mbs_idx` 等全部改为 `micro_batch` 全称，同步清理 assert 文案与单测。
- 静态路径硬 assert 的脆弱性（已解决）：guapisolo 评论 `The hard assert is required, but it's very fragile. There is no way to ensure len(step_micro_batches) % align_to == 0 or fit in target`，建议直接禁止非 multi-LoRA 路径混用 dynamic gbs；后续由 commit ee9e54d (#2197) 改为「启动即失败」并给出可操作报错，而非运行期静默出错。
- 后续清理：guapisolo 在 DISMISSED review 中说明 `recover some key cmts to improve readability. LGTM now, but better see #2135`，提示还存在一个后续清理 PR。
- 下游消费方：yushengsu-thu 在关联 Issue 下留言 `I need this pr for enabling variable global batch size in multi-lora #2137, waiting for this pr merging`，说明本 PR 是 multi-LoRA 可变 gbs 的前置依赖。
 - `dp_schedule` 中 `mbs` 标识符歧义 (style): guapisolo 亲自提交 commit f66b78a，将 `mbs/step_mbs/num_mbs/rank_mbs_idx` 等全部改为 `micro_batch` 全称，并同步清理 assert 文案与单测。
 - 静态路径对齐硬 assert 的脆弱性 (design): 后续 commit ee9e54d (#2197) 实现：静态路径无法对齐时启动即失败，并给出可操作报错 (`step_size % (dp_size * micro_batch_size * mb_group) == 0`)，避免运行期静默出错。
 - multi-LoRA 可变 gbs 的依赖关系 (question): 本 PR 将 multi-LoRA 明确排除在 rollout-side 调度之外（保留 legacy 路径），#2137 作为下游消费方等待合并。
 - 后续清理 PR #2135 (other): 存在一个后续清理 PR #2135 需跟进；本 PR 主体已获 APPROVED。

风险与影响

- 风险：具体风险点如下：
- 静态路径启动失败：`build_dp_schedule` 静态分支在 `step_size % (dp_size * micro_batch_size * mb_group) != 0` 时直接 `AssertionError`，老配置可能突然失败。这是刻意设计（启动即失败换取正确性），但团队需按报错提示调整配置。

- 记账重建依赖 all-reduce: rollout_mask_sums 分母广播到所有兄弟样本，靠现有 DP*CP all-reduce 把 sum/分母 重新拼出每 rollout 均值；任何新的切分维度（如未来 CP 扩展）都必须透传 denominator 一致性，否则数值偏差且难定位。
- TIS decoupled 语义变化：修改后的 response masks 只修正 numerator，denominator 仍用预计算的 rollout 总数；若 mask 修改面积很大，loss 会被系统性低估。
- metric 口径变化：reduce_gathered_log_dict 从 mean-of-rank-means 变成 count-weighted，(sum,count) 元组与旧 scalar 并存，跨版本对比指标时需注意口径不同。
- 性能：调度是纯 Python 且每 step 执行一次 Karmarkar-Karp / FFD，当样本量极大（数万 samples）时 CPU 耗时与内存占用可能成为 rollout 管理器的瓶颈。
- 数据契约扩展：ROLLOUT_DATA_VALUE_SPEC 新增 5 个键，老 trainer 与新版 rollout 混跑会出现 spec 不匹配；依赖 gbs 数值做显存估算的脚本需按 rollout 口径重新校准。
- 影响：影响范围与程度：
- 用户 / 实验者：新增 `--allow-partial-train-step`、`--balance-by-flops` 参数；compact/agent 负载的 loss 记账被修正（不再按样本数过加权）；拖尾 rollouts 可选训练而非丢弃。对经典「1 rollout = 1 sample」用户无感知（已有 30B 数值验证为严格 no-op）。
- 系统：调度职责从训练侧迁移到 rollout 侧，训练侧 get_data_iterator 简化（去掉 all_reduce(MAX) 同步点）；对象存储 spec 扩展 5 个键；指标日志从 per-rank mean 变为 (sum, count)。
- 团队：本 PR 是后续多特性的地基（multi-LoRA 可变 gbs #2137、fully-async 精确记账、dp-attention 调度验证），且改变了训练管线对 gbs 的语义理解，需要同步更新文档与实验脚本。
- 风险标记：核心训练管线重构，gbs 口径从样本改为 rollout，静态路径启动即失败，跨 DP/CP 记账依赖 all-reduce，TIS 解耦分母语义变化，指标归约口径变化

关联脉络

- PR #1673 [feat] support sample rollout submission granularity to keep fully async concurrency: 同属「rollout 记账粒度」演进线：fully-async 下样本数不定正是可变 gbs 的核心动机场景，本 PR 的 rollout_id 分组机制建立在其提交粒度重构之上。
- PR #1717 Rewrite fully-async rollout as FullyAsyncRolloutFn on the class-based rollout API: fully-async/agent 多轮 rollout 每次执行产出样本数不定，是 variable gbs 的动机来源；本 PR 的 per-rollout 记账补上了该路径的 loss 归一。
- PR #2041 Address engines by base URL when the router is dp-aware: 本 PR 新增 tests/e2e/megatron/test_qwen3_30B_A3B/test_dp_attention.py，验证 rollout-side 调度在 dp-aware 请求重排下保持顺序不敏感（按 rollout_id 分组），与 dp-attention 路由修复直接交叉。
- PR #2012 router: enable dp-aware routing under dp-attention: dp-aware 路由开启后样本完成顺序被打乱，本 PR 的调度必须不依赖完成顺序，二者构成同一 dp-attention x rollout 调度的验证链。