PR #2017 完整报告

THUDM/slime

feat: add --balance-by-flops for FLOPs-balanced micro-batching

合并时间：2026-06-06 23:01

原文链接：http://prhub.com.cn/THUDM/slime/pull/2017

执行摘要

- 一句话：引入 FLOPs 感知的微批次调度
- 推荐动作：值得精读，尤其是调度策略的权衡：token 平衡 vs FLOPs 平衡、近似公式 vs 精确计算。设计清晰，扩展性好。合并者计划将其作为默认调度策略，表明该设计符合长期演进方向。

功能与动机

动态批处理仅按 token 数平衡微批次（Σ L），但注意力 FLOPs 随序列长度二次增长。序列长度差异大时，少数长序列的 rank 计算成本远高于多数短序列的 rank，而 all-reduce 同步受最慢 rank 限制。--balance-by-flops 使用 coeff*L + L² 作为平衡指标，在微批次分区层面直接平衡计算负载，减少同步等待。

实现拆解

1. 参数声明与验证：在 slime/utils/arguments.py 中添加 --balance-by-flops 布尔参数，并在 slime_validate_args 中要求必须同时启用 --use-dynamic-batch-size。
2. FLOPs 工作量计算：在 slime/utils/seqlen_balancing.py 中新增 calculate_workload 函数（后来被 calculate_fwd_flops 替代），实现 coeff*L + L² 公式；同时在 dp_schedule.py 中新增 _calculate_workloads 辅助函数，调用精确的 calculate_fwd_flops 计算每个样本的 FLOPs。
3. 微批次打包路径变更：在 dp_schedule.py 的 _pack_step_into_mbs 中增加 balance_by_flops 分支——当启用时，先估算期望微批次数量，若每个样本可独占一个 mbs 则提前返回，否则计算 FLOPs 工作量并调用 get_seqlen_balanced_partitions 进行 Karmarkar-Karp 分区。
4. DP rank 分配优化：在 build_dp_schedule 中，当 balance_data 或 balance_by_flops 启用时，使用 KK 分配微批次到 rank；若 balance_by_flops 开启则使用 FLOPs 权重和代替 token 和进行分区。
5. 测试配套：初始未添加专门测试，但后续提交中更新了已有测试（test_dp_schedule.py）以适配新参数。

关键文件：

- slime/utils/dp_schedule.py（模块 调度器；类别 source；类型 core-logic；符号 _calculate_workloads, _pack_step_into_mbs, build_dp_schedule）：核心调度文件，修改

了微批次打包和 DP rank 分配逻辑，新增 FLOPs 平衡分支。

- slime/utils/seqlen_balancing.py（模块 序列长度平衡；类别 source；类型 core-logic；符号 calculate_workload）：新增 calculate_workload 函数，实现 FLOPs 估计公式 coeff*L + L²。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 core-logic）：新增 --balance-by-flops 命令行参数，并在 slime_validate_args 中验证依赖关系。

关键符号：_pack_step_into_mbs, build_dp_schedule, _calculate_workloads, calculate_workload, slime_validate_args

关键源码片段

slime/utils/dp_schedule.py

核心调度文件，修改了微批次打包和 DP rank 分配逻辑，新增 FLOPs 平衡分支。

```
def _pack_step_into_mbs(
    step_lengths: list[int],
    *,
    args: Any,
    use_dynamic_batch_size: bool,
    max_per_bin: int | None,
    micro_batch_size: int | None,
    balance_by_flops: bool = False,
) -> list[list[int]]:
    """Group a step's samples into mbs. Returns ``mbs[k]`` = local indices into ``step_lengths``.
    """
    if use_dynamic_batch_size:
        assert max_per_bin is not None
        if balance_by_flops:
            # 当启用 FLOPs 平衡时，计算期望的微批次数量，然后使用 Karmarkar-Karp 分区
            total_tokens = sum(step_lengths)
            num_mbs = max(1, (total_tokens + max_per_bin - 1) // max_per_bin)
            # 如果每个样本单独一个 mbs，直接返回
            if num_mbs >= len(step_lengths):
                return [[i] for i in range(len(step_lengths))]
            # 计算每个样本的 FLOPs 工作量
            workloads = _calculate_workloads(step_lengths, args)
            # 用 Karmarkar-Karp 按工作量分区，每个分区大小可不相等
            return get_seqlen_balanced_partitions(workloads, num_mbs, equal_size=False)
        # 默认的 token 数量 first-fit 打包
        return first_fit_pack(step_lengths, max_per_bin)
    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

- zhuzilin 建议复用精确 FLOPs 函数：建议使用已有的 *calculate_fwd_flops* 替代手工系数计算。HaoDong0027 进行了消融实验（Qwen3-30B-A3B，RL GRPO），对比基线（token-sum KK）、近似公式（*coeff*L+L²*）和精确 FLOPs，结果显示近似和精确方法性能相近（训练时间 -20.7% vs -21.2%），但精确方法在 token/s 上略优（+28.2% vs +23.9%）。最终采纳精确方法并移除了 *arguments.py* 中的手工逻辑。
- liujia-cc 关于微批次数量减少的疑问：指出 FLOPs 打包可能减少微批次数量，是否影响实验时间。实验数据表明训练时间显著减少，证明影响正面。
  - 使用精确 FLOPs 计算 vs 近似公式 (performance): 决定使用精确的 *calculate_fwd_flops* 函数替代近似公式，因为性能相近且更准确。
  - 微批次数量减少对实验时间的影响 (question): 权衡后认为减少微批次数量对训练时间无负面影响，因为性能提升数据证明了有效性。

风险与影响

- 风险：
  - 缺少测试覆盖：初始提交未包含专用测试，后续虽更新了 test_dp_schedule.py，但覆盖仍不充分，可能遗漏边界情况（如单样本超大、零样本等）。
  - FLOPs 估计误差：即使使用精确 *calculate_fwd_flops*，其依赖模型配置的准确性，若配置不正确可能导致次优平衡。
  - 与 --balance-data 交互：两者同时启用时，DP rank 分配优先使用 FLOPs 权重，与原有 token 平衡行为不一致，可能影响已有依赖 --balance-data 的工作流。
  - 默认关闭：对现有用户无影响，但新功能可能因未设置而未被充分利用。
- 影响：
  - 用户：通过可选标志获得训练加速，特别适用于序列长度分布不均匀的场景（如 RL 训练）。用户需了解 FLOPs 平衡的基本概念。
  - 系统：调度计算增加 FLOPs 估算开销（一次前向传播计算），但相对训练时间可忽略。
  - 团队：需维护新的参数和逻辑，后续 PR #2029 将其作为默认行为后，维护成本降低。
  - 风险标记：缺少测试覆盖，FLOPs 估计误差风险，与 --balance-data 交互未充分验证，默认关闭不影响现有行为

关联脉络

- PR #2029 *use balance_by_flops as balance data across mbs*: 后续 PR 将此功能作为默认行为，表明演进方向。
- PR #2028 *remove abundant function*: 移除了本 PR 引入的 *calculate_workload* 函数（后续重构中被替换）。